UNIVERSIDAD AUTÓNOMA DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación

TRABAJO FIN DE GRADO

EMULACIÓN EN TIEMPO REAL DE FUENTES DE ALIMENTACIÓN

Irene Villar Gómara
Tutor: Alberto Sánchez González
Ponente: Ángel de Castro Martín

Junio 2016

EMULACIÓN EN TIEMPO REAL DE FUENTES DE ALIMENTACIÓN

AUTOR: Irene Villar Gómara

TUTOR: Alberto Sánchez González

Hardware & Control Technology Laboratory

Departamento de Tecnología Electrónica y de las Comunicaciones

Escuela Politécnica Superior

Universidad Autónoma de Madrid

Junio de 2016

Resumen

El control digital para convertidores conmutados de potencia ha sufrido un gran crecimiento ya que ofrece beneficios significativos sobre el control analógico. La verificación de un regulador es clave antes de probarlo junto a una planta a regular. Sin embargo, la simulación de un regulador digital para convertidores conmutados no es trivial debido a la unión de un elemento analógico (planta) y otro digital (regulador). Además, la realización de pruebas con la planta real puede provocar accidentes debido a la gran cantidad de energía que es capaz de manejar un convertidor conmutado. Aunque hay soluciones que permiten esta verificación mediante simulación, no hay una metodología estándar, sino que en la literatura existen diferentes alternativas, y, en muchos casos, la verificación es un proceso lento debido a las herramientas que se usan.

Una de las opciones para la etapa de verificación es utilizar un sistema HIL (*Hardware-in-the-loop*) en el que se emula en hardware real un modelo digital de la planta con el mismo funcionamiento que el elemento a regular, generando simulaciones significativamente más rápidas que no pueden alcanzarse con otros medios de simulación convencionales. De esta manera, se mejora la etapa de pruebas del regulador y se evita tener que realizar pruebas sobre la planta real que pueda provocar daños.

El objetivo de este Trabajo Fin de Grado es implementar un modelo digital de un convertidor *full-bridge* utilizando aritmética en coma fija parametrizable, para integrarlo como sistema HIL. La implementación en coma fija, en vez de coma flotante, permite velocidades de emulación mayores, por lo que se podrá emular el modelo con precisión y en tiempo real, mientras que el área es muy baja. La gran desventaja de la coma fija es que está restringida a las condiciones de la simulación ya que la posición de la coma queda fijada durante la etapa de diseño y no puede cambiarse. Por ello, en este TFG se propone usar coma fija parametrizable, la cual puede ser configurada para cambiar la posición de la coma de las variables del modelo, de forma dinámica y sin necesidad de resintetizar código. Así, el modelo puede adaptarse a cualquier condición de simulación.

El trabajo muestra el diseño del modelo HIL, su integración en un sistema basado en microprocesador en una FPGA Xilinx Zynq, así como las pruebas que se realizan al modelo en lazo abierto con cada uno de los modelos presentados. En los resultados se muestra la precisión y los tiempos de simulación/emulación del modelo de diferentes aritméticas, incluida la coma fija parametrizable, así como los recursos utilizados en la FPGA.

Palabras clave

Hardware-in-the-loop, Emulación, Control digital, Fuentes de alimentación conmutadas.

Abstract

Digital control for switching power converters has undergone tremendous growth because it offers significant benefits over analog control. The verification of regulators is key to prove them together with the regulated plant. However, the digital regulator simulation for a power converter is not trivial due to the union between an analog element (plant) and a digital element (regulator). In addition to it, testing the system with a real plant can provoke accidents due to the big amount of energy which is capable to manage a power converter. Although there are solutions that allow this verification through simulation, there is not a standard methodology, but in literature there are different alternatives and, in many cases, the verification is a slow process due to the tools used.

One of the options for the verification stage is using an HIL system (*Hardware-in-the-loop*) in which a digital model of the plant with the same operation as the plant is emulated in real hardware, generating simulations significantly faster that cannot be reached with other conventional methods of simulation. This way, the testing stage is improved and testing with a real plant is avoided.

The objective of this Bachelor Thesis is to implement a digital model of a *full-bridge* converter using fixed point parametrizable arithmetic, to integrate it as a HIL system. The implementation in fixed point, instead of floating point, allows higher emulation speeds, so that it can emulate the model with accuracy and in real time, while the hardware area of the model is very low. The great disadvantage of the fixed point is that it is restricted to the simulation conditions because the position of the point is fixed during the design stage and it cannot be changed. For these reasons, in this work it is proposed to use parametrizable fixed point, which can be configured to change the position of the point of the model variables, dynamically and without resynthesizing code. Thus, the model can adapt itself to any simulation condition.

This work shows the design of the HIL model, its integration with a microprocessor-based system on a FPGA Xilinx Zynq, as well as the tests that have been realized to the model in open loop with each of the presented models. Results show the accuracy and the simulation/emulation time of the model of different arithmetic, including the parametrizable fixed point, as well as resources used on the FPGA.

Keywords

Hardware-in-the-loop, Emulation, Digital control, Switching-mode power supplies.

Agradecimientos

Quería agradecer a todas las personas que me han apoyado y ayudado en estos cinco años de carrera.

Gracias a mi tutor, Alberto, por guiarme en este trabajo, por darme esta gran oportunidad, por ayudarme, por estar día a día pendiente de mí y por todas y cada una de las correcciones de esta memoria.

Gracias a mis amigos, a los de toda la vida y a los que he podido ir descubriendo poco a poco durante estos años. Gracias por los viajes, las salidas, las charlas, los cotilleos, las merendolas… Simplemente, gracias por haber estado ahí y haberme hecho mucho más ameno el camino, sobretodo, este último año.

Por último, gracias a mis padres y a mis hermanas, por todo el apoyo, por aguantarme en mis más y en mis menos y por todo el tiempo dedicado para que esto haya sido posible. Gracias a mi familia en general, por el apoyo y los ánimos, a pesar de la distancia.

INDICE DE CONTENIDOS

INDICE DE FIGURAS

INDICE DE TABLAS

1 Introducción

1.1 Motivación

El control digital para convertidores conmutados ha sufrido un gran crecimiento ya que ofrece beneficios significativos sobre el control analógico [1]. Como cualquier otro sistema de regulación, éste debe ser probado después de implementarlo. Sin embargo, la verificación de un control digital junto a una planta analógica no es una tarea trivial, debida a la naturaleza mixta analógica-digital (figura 1).

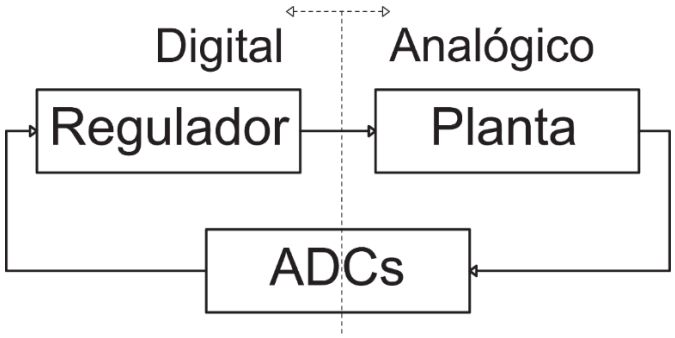


Figura 1. Esquema analógico-digital de un convertidor de potencia.

El control digital presenta otros inconvenientes que hacen que se sigan utilizando controles analógicos, entre ellos, el coste. Los elementos utilizados para el control digital son mucho más baratos, es por eso por lo que hay que buscar con el control digital alguna ventaja que permita reducir el precio del sistema. Pero, con el gran crecimiento de lo digital, ya se pueden encontrar algunos controles que pueden competir con el precio de lo analógico.

Por otro lado, en control digital se trabaja con variables analógicas que hay que digitalizar para que puedan ser usadas, lo cual genera otro inconveniente en el control digital. Es por eso por lo que es necesario el uso de convertidores DC/AC y AC/DC.

Aún con los inconvenientes mencionados, el control digital tiene muchas otras ventajas que compensa estos inconvenientes. Aunque el control analógico se siga usando, por ejemplo, en el caso de trabajos sencillos, cada vez se pueden encontrar más convertidores digitales tanto en ámbitos de investigación como comerciales.

Por otra parte, la realización de pruebas en el hardware real puede provocar accidentes debido a la gran energía que es capaz de manejar un convertidor conmutado. Por tanto, para el desarrollo de un control digital que regula una planta analógica, son necesarias simulaciones previas para comprobar su correcto funcionamiento. Sin embargo, no hay una metodología estándar para realizar estas pruebas, sino que en la literatura se encuentran diferentes alternativas. A la hora de diseñar el regulador digital, se pueden utilizar herramientas como Matlab para comprobar la estabilidad, la respuesta, etc., pero cuando se implementa el regulador en hardware o software, pueden cometerse errores de implementación, difíciles de detectar. Por ejemplo, se puede implementar un código erróneo, se pueden realizar una comunicación con fallos con los ADCs, puede segmentarse mal el

código en el caso de implementaciones basadas en FPGA, etc. Como se verá en el Estado del arte, existen diferentes estrategias de simulación, entre las que destaca HIL (*Hardware-in-the-loop*).

HIL consiste en sustituir el elemento a regular o planta por un modelo que tenga el mismo comportamiento y ejecutarlo en hardware en tiempo real, de tal manera que el regulador no note si lo que está controlando sea una planta real o el modelo HIL. Esta metodología de simulación es muy utilizada en el sector aeroespacial y automovilístico para la prueba de unidades de control o simulación de fallos. También se usa habitualmente para la simulación de motores y redes eléctricas y, últimamente, también se utiliza para la simulación de convertidores conmutados de potencia.

1.2 Objetivos

Este Trabajo Fin de Grado tiene como objetivo principal realizar un modelo en VHDL en coma fija de un convertidor conmutado, en particular el modelo de una topología *full-bridge*, para su emulación HIL. Para ello se han considerado los siguientes requisitos:

- Realizar un modelo parametrizable para que sea útil bajo diversas condiciones de carga, tensiones, frecuencias de conmutación, etc. Es decir, evitar el problema clásico de los modelos en coma fija en los que hay unos límites rígidos en el rango de representación numérica de las señales que modelan el convertidor.
- Como objetivo secundario de este TFG se realizarán modelos en coma flotante no sintetizable, coma flotante sintetizable y coma fija no parametrizable para realizar una comparativa de precisión y velocidad entre las diferentes alternativas.
- Con el objetivo de conseguir un modelo HIL útil, se requerirá que la información calculada por el modelo (tensiones y corrientes) tenga salidas analógicas, de modo que un supuesto sistema de control pueda conectarse al modelo HIL de igual forma a como se conectaría a una planta real. Es decir, se conseguirá que el sistema de control no se deba modificar para poder ser probado junto al modelo HIL propuesto en este TFG.
- La parametrización y configuración del modelo HIL deberá poder realizarse en tiempo de ejecución y mediante un puerto de comunicación serie. Aprovechando esta interfaz, se requerirá que información adicional a las variables de estado se envíe a un ordenador. Por ejemplo, podrán establecerse alarmas de sobretensión, sobrecorriente, también requisitos de tensiones y corrientes máximas y mínimas, etc. Para esta tarea se requerirá que el modelo HIL esté integrado en un sistema basado en procesador, ya que aportará gran flexibilidad a las comunicaciones y a la gestión de alarmas.

1.3 Organización de la memoria

Este trabajo fin de grado se estructura de la siguiente manera:

- El segundo capítulo presenta el estado del arte de este trabajo que detalla las diferentes alternativas de simulación en la literatura para comprobar el correcto funcionamiento de un control digital que regula una planta analógica.
- El tercero capítulo detalla cómo crear un modelo del convertidor *full-bridge* obteniendo las ecuaciones de las variables de estado y, con ello, la función de transferencia de la planta.
- El cuarto capítulo explica los detalles de las distintas posibilidades de implementación.
- El quinto capítulo presenta las comparaciones realizadas entre los distintos modelos implementados, tiempos de simulación/emulación y los recursos usados en los diseños.
- Por último, el sexto capítulo presenta las conclusiones finales de este trabajo así como el posible trabajo futuro.

2 Estado del arte

Como se ha mencionado en el apartado 1.1, en la literatura se encuentran diferentes alternativas para la realización de simulaciones previas para comprobar el correcto funcionamiento de un control digital que regula una planta analógica.

Una opción clásica para la depuración sería utilizar un simulador mixto digital-analógico, para poder simular la planta y el control simultáneamente [2, 3, 4]. En [4] se utiliza un simulador mixto para un sistema cuyo convertidor está descrito en Spice, el regulador está implementado en VHDL sintetizable y los ADCs están modelados en VHDL-AMS. La figura 2 se extrae de [4].

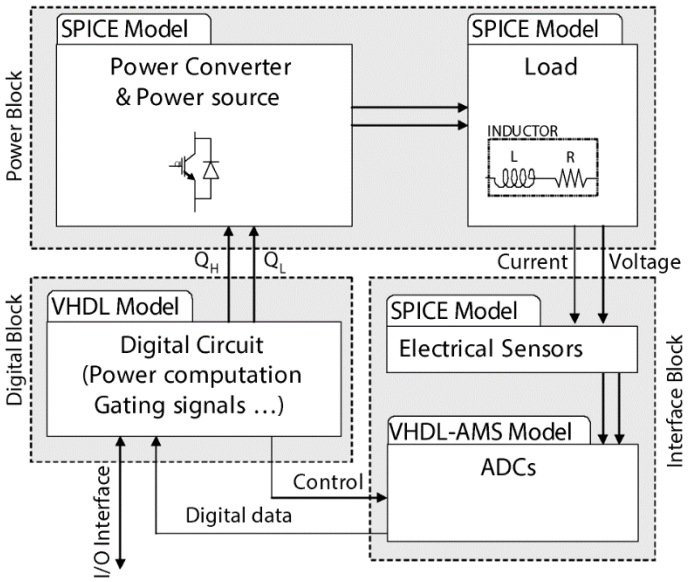


Figura 2. Simulación mixta con señales analógicas/digitales. Imagen extraída de [4].

Sin embargo, el principal problema de estos simuladores es que la velocidad de simulación de estas herramientas es muy lenta. Por otro lado, la existencia de estos simuladores mixtos en el mercado no es tan accesible como otras alternativas porque no hay muchos simuladores disponibles y por su precio. Dependiendo del producto elegido, el simulador puede interpretar lenguaje C, VHDL [5, 6], o incluso Java [7].

Otra solución propuesta en la literatura es utilizar dos simuladores, uno para el control y otro para la planta, y desarrollar una interfaz personalizada de comunicación entre simuladores [8].

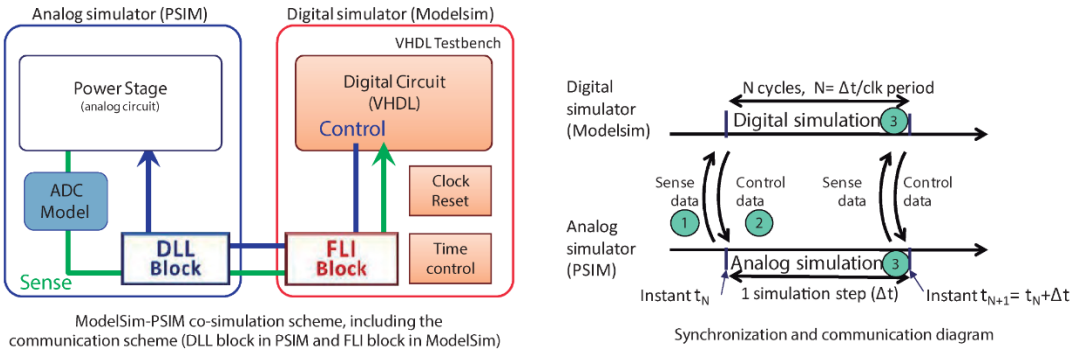


Figura 3. Simulación PSIM-ModelSim. Imagen extraída de [8].

Otra alternativa propuesta es diseñar un modelo digital de la planta en un lenguaje HDL (*Hardware Description Language*), lo que permite simulaciones más rápidas [9, 10, 11]. Al realizar este diseño se puede simplificar ligeramente el modelo, ya que la finalidad de esta etapa de verificación no es tanto medir la estabilidad del sistema y su respuesta, sino comprobar que la implementación final del control es correcta. En el caso de un regulador digital escrito en HDL, se puede implementar el modelo y el regulador en una FPGA (*Field-Programmable Gate Array*), creando un sistema de emulación HIL (*Hardware-in-the-loop*). Es decir, en vez de simular en software el modelo, se implementa dicho modelo en hardware para acelerar su ejecución. Incluso si el control digital no está diseñado en HDL, sino en un microprocesador o DSP (*Digital Signal Processor*), se puede emular el modelo de la planta en una FPGA e interconectar las entradas y salidas del modelo HIL al dispositivo regulador. El uso de FPGAs para modelar convertidores conmutados permite reducir el mínimo paso de integración posible en la simulación, por lo que permite simulaciones más precisas [12, 13].

En la literatura se encuentran modelos HDL para convertidores conmutados en coma flotante y en coma fija. Los modelos en coma flotante de 32 bits son fáciles de implementar [14, 15]. En [16] se ve que un modelo en coma flotante de 32 bits puede no tener suficiente resolución si el regulador tiene una frecuencia de conmutación alta (centenas de kHz). Esto es debido a que frecuencias altas de conmutación requieren que el paso de integración sea muy pequeño y, por tanto, los incrementos en las variables de estado serán también pequeños, por lo que la resolución se convierte en un problema. Se pueden usar más bits para aumentar la resolución usando coma flotante de doble precisión o tamaños no estándares, pero el área ocupada aumenta significativamente y la frecuencia de trabajo del modelo se ve reducida.

Otra opción es diseñar un modelo en coma fija, asignando el número de bits necesarios para cada variable del modelo, y optimizando área y frecuencia de trabajo, pero aumentando el esfuerzo de diseño [16]. Además, otro problema de los modelos en coma fija presentados en la literatura es que están definidos para unos rangos de valores fijos, es decir, sólo funcionan correctamente hasta unos niveles máximos de tensión, corriente, frecuencia de conmutación, etc. Esta restricción a la hora de diseñar, que en coma flotante no existe, puede paliarse si se realiza un modelo en coma fija parametrizable, donde el número de bits dedicados a la parte entera y parte fraccionaria sea variable. No se han encontrado propuestas en la literatura sobre el uso de coma fija parametrizable en tiempo de ejecución, al menos en la búsqueda que se ha realizado para emulación de convertidores conmutados.

3 Modelado del convertidor

Este trabajo presenta la implementación de un modelo digital y parametrizable en coma fija de un convertidor conmutado. Como ejemplo de aplicación, se va a diseñar el modelo de una topología *full-bridge*. Aunque se ha elegido esta topología, la metodología usada en este TFG se puede aplicar a cualquier otro convertidor conmutado.

3.1 Posibilidades de implementación

Como se vio en el Estado del arte, las FPGAs aportan ventajas en la simulación HIL de convertidores de potencia. Si se opta por usar una FPGA y modelar la planta utilizando un lenguaje HDL, se pueden destacar cuatro posibilidades de implementación del modelo en función de la aritmética usada:

1. Modelo en coma flotante no sintetizable (utilizando señales de tipo *real* en el caso de VHDL): Es un modelo fácil de implementar ya que las señales ajustan automáticamente su exponente consiguiendo así la mejor resolución posible. Está soportado por la mayoría de los simuladores. Sin embargo, no puede ser sintetizado. El tipo de datos *real* utiliza el estándar IEEE 754 de doble precisión (53 bits de mantisa), por lo que la resolución obtenida permite sobradamente el modelado de una planta incluso con frecuencias de conmutación elevadas. Aunque es la aritmética más directa de usar y sus similitudes con la programación software son elevadas, no se puede considerar para un sistema HIL ya que no puede ser implementada en hardware real al no ser sintetizable.
2. Modelo en coma flotante sintetizable: Existen bibliotecas como *float* en el estándar VHDL-2008 que implementan este estándar. El estándar IEEE 754 propone dos formatos, precisión simple y doble precisión, donde la última exige una cantidad de recursos no asumible en muchos casos. En la literatura se encuentran modelos de convertidores usando coma flotante sintetizable de precisión simple [9], pero esta aritmética puede no tener suficiente resolución si la frecuencia de conmutación es alta y por tanto no siempre es viable en el modelado de convertidores conmutados. También es posible usar formatos de coma flotante con un tamaño de mantisa intermedio entre la precisión simple y la precisión doble. Sin embargo, los recursos usados por cualquier sistema de coma flotante son muy elevados mientras que la frecuencia de conmutación es mucho menor que usando coma fija.
3. Modelo en coma fija con librería *sfixed*: La coma fija permite obtener frecuencias de síntesis más altas utilizando menos recursos, pero con un esfuerzo de diseño mucho mayor. La librería *sfixed* de VHDL-2008 facilita esta tarea interpretando los resultados en coma fija con sus respectivos tamaños y facilitando las operaciones aritméticas. Este modelado es sintetizable, pero debe ser diseñado teniendo en cuenta los rangos de valores que tendrá cada señal, quedando fijado, por tanto, el número de bits de parte entera y de parte fraccionaria y, consecuentemente, la máxima resolución. Por tanto, cambios en los rangos de valores reales durante la ejecución del modelo pueden hacer que la resolución sea insuficiente o que los valores desborden la capacidad total de una señal. Si esto ocurriera, debe rediseñarse el modelo cambiando los tamaños de las señales del

modelo o, al menos, reajustando el número de bits de parte entera y fraccionaria. Por todo eso, la complejidad de este modelo está en la elección de los tamaños de las señales.

4. Modelo en coma fija parametrizable: Se utilizan los fundamentos de la coma fija, pero sin utilizar ninguna biblioteca ya que a priori no se sabe el número de bits dedicados a las partes entera y fraccionaria. En el caso de VHDL se utiliza señales de tipo *std_logic_vector*, que no interpretan dónde está la coma, ni facilitan las operaciones aritméticas. Por tanto, las entradas y las salidas del modelo deben ser interpretadas externamente. Además, el modelo se realiza parametrizable de forma que, en vez de haber un número estático de decimales en cada señal, la posición de la coma puede cambiar de forma dinámica según la configuración de la simulación, y sin necesidad de resintetizar. Es decir, se consigue un modelo intermedio entre la coma flotante y la coma fija. Así, se trata de un modelo sintetizable, se pueden elegir los bits necesarios para cada señal de forma individual y la resolución siempre es óptima para toda simulación.

El objetivo en este TFG es la implementación de un modelo parametrizable de un convertidor *full-bridge*, esto es, hacer un modelo genérico que funcione para cualquier condición de simulación.

3.2 Modelado de un full-bridge

La figura 4 muestra la topología básica de un *full-bridge*. Está compuesto por cuatro interruptores (MOSFETs en este caso) con un diodo en antiparalelo en cada interruptor, dos elementos pasivos que son una bobina y un condensador y una carga que en este caso se ha considerado resistiva, aunque otro tipo de carga se puede modelar de forma similar.

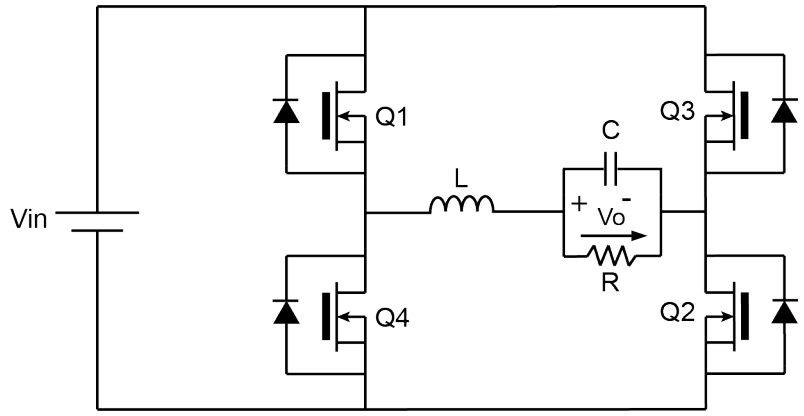


Figura 4. Esquemático de un puente en H o *full-bridge*.

Se trata de un circuito que puede actuar como inversor en función de la secuencia de apertura y cierre de los interruptores. El modelo debe calcular en cada instante sus variables de estado, es decir, las señales que no pueden cambiar de valor de forma arbitraria en tiempo nulo. Las variables de estado en este modelo son la corriente que atraviesa la bobina i_L y la tensión de salida del circuito v_{out}.

La tensión de la bobina v_L viene definida por (3.1)

$$v_L = L \cdot \frac{di}{dt} \tag{3.1}$$

Transformando (3.1) en una ecuación en diferencias, la corriente de la bobina viene definida por (3.2)

$$i_L(k) = i_L(k-1) + \frac{\Delta t}{L} \cdot v_L \tag{3.2}$$

Donde $i_L(k)$ es la corriente que atraviesa la bobina en el instante k, por lo que $k-1$ es el instante anterior, Δt es el paso de integración para calcular las variables de estado, L es la inductancia de la bobina.

De la misma manera, la corriente que atraviesa el condensador se va a definir como (3.3):

$$i_C = C \cdot \frac{dv}{dt} \tag{3.3}$$

Esta ecuación también se puede transformar en una ecuación en diferencias (3.4):

$$v_{out}(k) = v_{out}(k-1) + \frac{\Delta t}{C} \cdot i_C \tag{3.4}$$

Siendo $v_{out}(k)$ la tensión del condensador en el instante de tiempo k, C la capacidad del condensador.

Una vez definidas las ecuaciones de las variables de estado, se van a distinguir diferentes casos en función del estado de cada uno de los interruptores (figura 5 y figura 6).

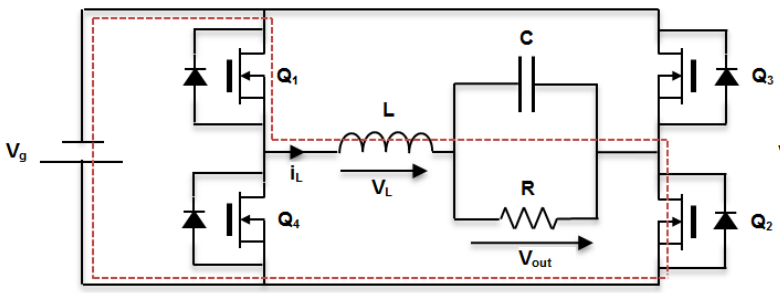


Figura 5. Rama 1 de corriente en un convertidor inversor.

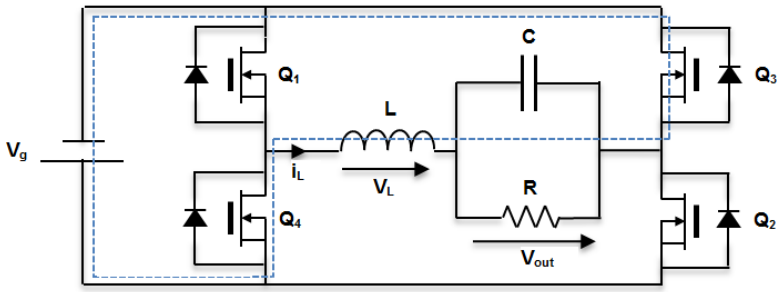


Figura 6. Rama 2 de corriente en un convertidor inversor.

Cuando los interruptores Q1 y Q2 están cerrados y Q3 y Q4 abiertos (3.5) se denomina rama 1, en la que la tensión de la bobina será la diferencia entre la tensión de entrada y la de salida $v_L = v_g - v_{out}$. En el caso contrario (3.6), Q3 y Q4 cerrados y Q1 y Q2 abiertos, siendo esta la rama 2, la tensión de la bobina es igual pero la tensión de entrada es negativa $v_L = -v_g - v_{out}$. Para el cálculo de la tensión de salida no afecta la tensión de entrada, por lo que, independientemente del estado de los interruptores, dependerá únicamente de la corriente que atraviesa el condensador $i_C = i_L - i_R$.

$$i_L(k) = i_L(k-1) + \frac{\Delta t}{L} \cdot \left(v_g - v_{out}\right)$$

$$v_{out}(k) = v_{out}(k-1) + \frac{\Delta t}{C} \cdot (i_L - i_R) \tag{3.5}$$

$$i_L(k) = i_L(k-1) + \frac{\Delta t}{L} \cdot \left(-v_g - v_{out}\right)$$

$$v_{out}(k) = v_{out}(k-1) + \frac{\Delta t}{C} \cdot (i_L - i_R) \tag{3.6}$$

Ambas ramas no pueden estar activas simultáneamente ya que se produciría un cortocircuito, así como tampoco podrían estar activos combinaciones de interruptores de dos ramas diferentes. Para evitar que se produzcan cortocircuitos debido a los retrasos de conmutación, es habitual añadir en el control un tiempo de transición entre la activación de una rama a otra, llamado *deadtime* o tiempo muerto, en el que ambas ramas se desactivan. Esto es debido a que los retrasos de conmutación de cierre a apertura y viceversa no son iguales, y pueden variar según los niveles de tensión, corrientes, temperatura, etc., que se estén manejando. Cuando todos los interruptores están cerrados, la corriente que debería circular por los interruptores atraviesa los diodos en antiparalelo. En este caso, cuando $i_L > 0$ sus ecuaciones son las correspondientes a la rama 2 (3.6), y cuando $i_L < 0$ sus ecuaciones son las correspondientes a la rama 1 (3.5).

3.3 Función de transferencia

El modelo que se implementará usará las ecuaciones descritas en el apartado anterior. Sin embargo, para probar que realmente el modelo se comporta como se espera, se va a calcular su función de transferencia y se va a comparar la dinámica y las ganancias de dicha función con el modelo. Asumiendo que el convertidor opera en régimen permanente, se puede extraer la tensión de salida que tendrá el convertidor en función del ciclo de trabajo y la dinámica de la planta cuando se modifica dicho ciclo de trabajo.

Partiendo de la tensión de la bobina (3.1) y siendo Δi_{L1} la variación de la corriente de la bobina cuando está activa la rama 1, es decir, cuando los interruptores Q1 y Q2 están cerrados y Q3 y Q4 abiertos, se obtiene 3.7.

$$\Delta i_{L1} = \left(v_g - v_{out}\right) \cdot \frac{d \cdot T}{L} \tag{3.7}$$

Siendo d el ciclo de trabajo expresado en tanto por uno y T el periodo de conmutación.

Por otro lado, se define Δi_{L2} como la variación de la corriente de la bobina cuando está activa la rama 2, es decir, el caso contrario que en Δi_{L1}, cuando están los interruptores Q3 y Q4 cerrados y Q1 y Q2 abiertos (3.8).

$$\Delta i_{L2} = \left(-v_g - v_{out}\right) \cdot \frac{(1-d) \cdot T}{L} \tag{3.8}$$

La variación de la corriente que atraviesa la bobina tiene que ser igual en ambas partes del ciclo de trabajo (rama 1 y rama 2) y de signo opuesto. Es decir, la corriente de la bobina debe tener el mismo valor al inicio y al final del ciclo de conmutación (3.9) para que se cumpla la suposición de régimen permanente.

$$\Delta i_{L1} + \Delta i_{L2} = 0 \tag{3.9}$$

$$(v_g - v_{out}) \cdot \frac{d \cdot T}{L} + (-v_g - v_{out}) \cdot \frac{(1-d) \cdot T}{L} = 0 \tag{3.10}$$

$$(v_g - v_{out}) \cdot d + (-v_g - v_{out}) \cdot (1-d) = 0 \tag{3.11}$$

Desarrollando la ecuación 3.11, se obtiene el valor de la tensión de salida en régimen permanente (3.12).

$$\frac{v_{out}}{v_g} = 2 \cdot d - 1 \tag{3.12}$$

La ecuación anterior expresa la ganancia del convertidor en función del ciclo de trabajo y la tensión de entrada. Sin embargo, expresa valores promedios y, por tanto, no especifica la dinámica del circuito. Para calcular la dinámica del circuito se puede extraer la función de transferencia de la planta. Para ello, se puede obtener un modelo en pequeña señal. Utilizando un modelo en pequeña señal se consideran sólo las perturbaciones de las variables

del modelo en torno a un punto de equilibrio. Dentro de la notación que se va a utilizar para el modelo en pequeña señal, $\langle x \rangle$ se denomina al promedio móvil durante un periodo de conmutación de una supuesta señal llamada x, el cual se define como $\langle x \rangle = X + \hat{x}$ siendo X el punto de operación en un instante dado y $\hat{x}$ una pequeña perturbación alrededor de él.

Para el modelo en pequeña señal se parte del circuito equivalente de un *full-bridge* de la figura 7 [17].

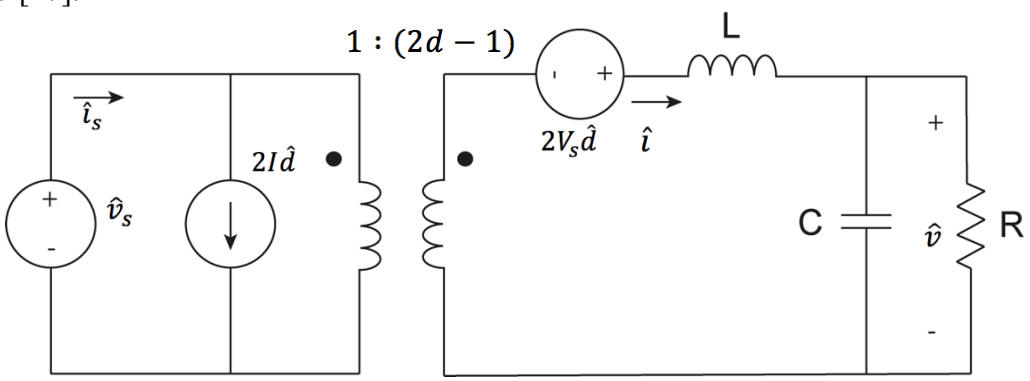


Figura 7. Circuito equivalente del modelo de pequeña señal para un *full-bridge*.

Analizando el circuito equivalente, se plantean las siguientes ecuaciones teniendo en cuenta los elementos pasivos (condensador y bobina):

$$L \cdot \frac{\partial \langle i \rangle}{\partial t} = (2 \cdot \langle d \rangle - 1) \cdot V_g - \langle v_{out} \rangle$$

$$C \cdot \frac{\partial \langle v_{out} \rangle}{\partial t} = \langle i \rangle - \frac{\langle v_{out} \rangle}{R} \tag{3.13}$$

Se pretende conseguir la función de transferencia del circuito para la tensión de salida en función del ciclo de trabajo, mientras que la tensión de entrada es constante. Por tanto, se considera que V_g es una constante y, por tanto, no utiliza la notación de pequeña señal:

$$\langle i \rangle = I + \hat{\imath}$$

$$\langle v_{out} \rangle = v_{out} + \hat{v}_{out}$$

$$\langle d \rangle = D + \hat{d} \tag{3.14}$$

Sustituyendo (3.14) en (3.13), se obtienen las siguientes ecuaciones.

$$L \cdot \frac{\partial (I + \hat{\imath})}{\partial t} = \left(2 \cdot \left(D + \hat{d}\right) - 1\right) \cdot V_g - (v_{out} + \hat{v}_{out})$$

$$C \cdot \frac{\partial (v_{out} + \hat{v}_{out})}{\partial t} = (I + \hat{\imath}) - \frac{(v_{out} + \hat{v}_{out})}{R}$$

A partir de las ecuaciones anteriores y teniendo sólo en cuenta las perturbaciones, se obtienen las siguientes ecuaciones.

$$L \cdot \frac{\partial \hat{\imath}}{\partial t} = 2 \cdot \hat{d} \cdot V_g - \hat{v}_{out} \tag{3.15}$$

$$C \cdot \frac{\partial \hat{v}_{out}}{\partial t} = \hat{\imath} - \frac{\hat{v}_{out}}{R} \tag{3.16}$$

Despejando la corriente de la ecuación (3.16) y sustituyéndola en (3.15), se obtiene:

$$L \cdot C \cdot \ddot{v}_{out} + L \cdot \frac{\dot{v}_{out}}{R} - \hat{v}_{out} = 2 \cdot \hat{d} \cdot V_g \tag{3.17}$$

Donde $\dot{v}_{out}$ representa la primera derivada de v_{out} y $\ddot{v}_{out}$ la segunda derivada. Por último, a la ecuación (3.16) se aplica Laplace y se obtiene la función de transferencia del *full-bridge*.

$$\left(LCs^2 + \frac{L}{R}s + 1\right) V_{out} = 2dV_g$$

$$\frac{v_{out}(s)}{d(s)} = \frac{2V_g}{LCs^2 + \frac{L}{R}s + 1}$$

Tal y como se esperaba, la función de transferencia expresa un segundo orden, ya que hay una bobina y un condensador. Por otro lado, la ganancia también es la esperada, porque con d = 0, se tiene $v_{out} = -v_g$ y con d = 1 se tiene $v_{out} = v_g$. La función de transferencia dice que un escalón unidad en ciclo de trabajo hace variar v_{out} 2 veces v_g, por lo que coincide.

4 Implementación

4.1 Introducción

En esta sección se describe el proceso de implementación de los modelos descritos en el apartado 3.1. Aunque el objetivo principal es el modelo en coma fija parametrizable, se describen el resto de modelos para poder entender los diferentes métodos y entender el proceso de diseño. Para su implementación, se utilizará el lenguaje VHDL, aunque su implementación en otros lenguajes HDL serían similares.

4.2 Modelo en coma flotante no sintetizable

El modelo en el tipo de datos *real* es el más fácil de implementar. Para su implementación no es necesario un esfuerzo extra por parte del diseñador para conseguir una resolución óptima ya que estas señales implementan coma flotante IEEE 754 de doble precisión (64 bits) y el ajuste de la coma se hace automáticamente. Pero como se ha visto anteriormente, aunque esté soportado por la mayoría de los simuladores, no es sintetizable. Debido a su gran resolución y, por tanto, la exactitud de sus resultados, se considerará como referencia para compararlo con el modelo en coma fija parametrizable.

Para la implementación del modelo con señales de tipo *real*, se han empleado las ecuaciones descritas en (3.5) y (3.6), las cuales simplemente se transcribirán en código VHDL. La declaración de las señales de tipo *real* se hace según la figura 8. Dichas ecuaciones que calculan las variables de estado irán dentro de un proceso síncrono, figura 9, en el cual se van a ir actualizando las variables de estado del modelo. El tiempo de integración Delta se corresponde al periodo de reloj de dicho proceso síncrono. Por otro lado, deltaL y deltaC en el código, se corresponden a Delta/L y Delta/C los cuales se precalculan para que se ahorre la división en código. Aunque en *real* esta simplificación no es importante, ahorra recursos y mejora la frecuencia en las versiones sintetizables descritas a continuación.

```
constant inc : real := 0.000000023; --23 ns
constant C : real := 0.0001; --0.1 mF
constant L : real := 0.0009; --0.9 mH
constant deltaC : real := inc/C;
constant deltaL : real := inc/L;

signal iLAux, iLAdd : real;
signal VoutAux : real;
signal iinAux : real;
```

Figura 8. Declaración de señales de tipo *real*.

```
process (CLK, Reset)
begin
     if Reset = '1' then
          VoutAux <= 0.0;
          iLAux <= 0.0;
     elsif clk'event and clk='1' then
          iLAux <= iLAux + deltaL * iLAdd;
          VoutAux <= VoutAux + deltaC * (iLAux - iR);
```

```
        end if;
    end process;
```

Figura 9. Cálculo de las variables de estado. Modelo *real* del *full-bridge.*

Por otro lado, habrá otro proceso en el que se calcule el valor de iLAdd, figura 10, siendo este el multiplexor que da valor a la tensión de la bobina en función de los estados de los cuatro interruptores.

```
process (dq, Vg, VoutAux)
begin
     case dq is
          when "01" =>
               iLAdd <= (-Vg - VoutAux);
          when "10" =>
               iLAdd <= (Vg - VoutAux);
          when others =>
               iLAdd <= (-VoutAux);
     end case;
end process;
```

Figura 10. Asignación de señales al multiplexor de corriente. Modelo *real* del *full-bridge.*

Donde *dq* es el valor de las distintas combinaciones de los estados de los interruptores, correspondiéndose con la figura 11.

```
d <= '1' when (Q1='1' or (Q1='0' and Q3='0' and iLAux <= 0.0)) else
     '0' when (Q3='1' or (Q1='0' and Q3='0' and iLAux > 0.0));
q <= '1' when (Q4='1' or (Q4='0' and Q2='0' and iLAux > 0.0)) else
     '0' when (Q2='1' or (Q4='0' and Q2='0' and iLAux <= 0.0));

dq <= (d & q);
```

Figura 11. Combinaciones de los estados de los interruptores. Modelo *real* del *full-bridge.*

4.3 Modelo en coma flotante sintetizable

El modelo en coma flotante sintetizable descrito en este apartado utiliza la librería *float_pkg* de VHDL-2008. Este modelo es muy parecido al modelo en el tipo de datos *real* descrito anteriormente. Dicha librería define los tipos *float32* y *float64*, sin embargo, en este trabajo se ha utilizado el tipo de 32 bits debido a la gran cantidad de recursos no asumibles que exige el tipo de 64 bits. Aun así, los recursos utilizados en 32 bits siguen siendo todavía muy elevados, como se verá en el capítulo de resultados.

El modelo en coma flotante sintetizable es muy parecido al modelo en coma flotante no sintetizable. La única diferencia en el código de ambos modelos es la declaración de las señales, correspondiéndose ahora con la figura 12.

```
constant inc : float32 := to_float(0.000000023);
constant C : float32 := to_float(0.000100);
constant L : float32 := to_float(0.000900);
constant deltaL : float32 := inc/L;
constant deltaC : float32 := inc/C;

signal iLAux, iLAdd : float32;
```

```
signal VoutAux : float32;
signal iinAux : float32;
```

Figura 12. Declaración de señales de tipo *float32*.

Los dos procesos de los que consta el modelo serán, por lo tanto, los correspondientes a los de las figuras 9 y 10, siendo las combinaciones de los estados de los interruptores las de la figura 11. La única diferencia es que las asignaciones de ceros, `0.0`, hay que pasarlos a tipo *float32*: `to_float (0.0)`.

Tal y como se indicó en la Introducción, este modelo puede tener problemas de resolución si el tiempo de integración es pequeño. Esto es debido a que el tipo *float32* utiliza señales de 32 bits, las cuales no tienen suficiente resolución ya que usa 24 bits de mantisa, de los cuales hay un "1" fijo y 23 bits adicionales. Así, para un valor de tensión de salida en torno a los 200 V, una corriente de bobina de 2 A y una corriente de carga de 1,8 A, partiendo de los datos declarados de la figura J, el valor de la tensión de salida en el siguiente ciclo de integración es:

$$v_{out} = v_{out} + (i_L - i_R) \cdot \frac{\Delta t}{C} = 200 + 0{,}000046 = 200{,}000046\ V$$

El valor 200,000046 debe ser representado en *float32,* pero el valor más aproximado y representable en este formato sería 200,00005 V, por lo que habría un error absoluto de 0,00004 V. Aunque este error parece despreciable, se va acumulando en cada ciclo de reloj haciendo que cada vez el modelo vaya siendo menos preciso. Como se verá en el capítulo de Resultados, el problema de resolución en float32 no permitirá que se puede utilizar esta aritmética para el sistema HIL propuesto.

4.4 Modelo en coma fija con librería sfixed

Se va a emplear la librería *sfixed* de VHDL, la cual permite usar aritmética en coma fija y además realiza automáticamente el redondeo y saturación para evitar desbordamientos de los resultados de operaciones aritméticas. Sin embargo, la complejidad de este modelo está en la elección de los tamaños de las señales, ya que estos deben ser fijados en tiempo de diseño.

El modelo en coma fija utiliza la notación QX.Y. Este formato, como se puede ver en la figura 13, contiene 1 bit que permite signo, X bits de parte entera, Y bits de parte fraccionaria, considerándose escrito en complemento a 2, por lo que una señal Q2.3 tendrá 6 bits. Para obtener el valor decimal de un número en binario en QX.Y hay que tener en cuenta la escala de la señal, siendo esta la cantidad de bits que tiene la parte fraccionaria. Así, si se traduce una señal binaria en formato QX.Y a decimal habría que multiplicar la señal por 2^{-Y}. Por ejemplo, si se tiene una señal que representa una tensión de salida y su valor binario es 101001011010000_2 (21200 en decimal antes de aplicar la escala) con escala igual a 11, la tensión tendrá un valor de $21200 \cdot 2^{-11} = 10{,}3515625$ V.

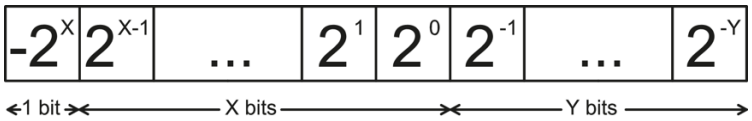


Figura 13. Formato de una señal en QX.Y.

En coma fija con librería *sfixed*, las señales se declaran como se muestra en la figura 14. Esta vez las ecuaciones no se transcriben tal cual, sino que se calculan haciendo las operaciones por separado redimensionándolas si fuera necesario. Finalmente, las variables de estado se actualizan en un proceso, siendo posteriormente redimensionadas y convertidas al formato de salida de las señales (figura 15). De la misma manera que en los modelos anteriores, se calcula el valor de iLAdd, figura 16.

```
signal deltaC : sfixed(-11 downto -35); --Q-11.35
signal deltaL : sfixed(-14 downto -38); --Q-14.38

signal VgAux : sfixed(8 downto -8); --Q8.8
signal iRAux : sfixed(4 downto -12); --Q4.12
signal VoutAux, VoutAux_sat : sfixed(8 downto -47); --Q8.47
signal VoutAux_redim : sfixed(8 downto -8); --Q8.8
signal iLAux, iLAux_sat : sfixed(6 downto -46); --Q6.46
signal iLAux_redim : sfixed(4 downto -12); --Q4.12

signal iLAdd : sfixed(9 downto -8); --Q9.8
signal VoutAdd : sfixed(5 downto -12); --Q5.12
signal mult1 : sfixed(-4 downto -46); --Q-4.46
signal mult2 : sfixed(-5 downto -47); --Q-5.47
```

Figura 14. Declaración de señales con librería *sfixed*.

```
mult1 <= resize (iLAdd*deltaL, mult1);
VoutAdd <= iLAux_redim - iRAux;
mult2 <= resize (VoutAdd*deltaC, mult2);

iLAux_sat <= resize (iLAux + mult1, iLAux_sat);
VoutAux_sat <= resize (VoutAux + mult2, VoutAux_sat);

    process (CLK, Reset)
    begin
        if Reset = '1' then
                VoutAux <= (others => '0');
                iLAux <= (others => '0');
        elsif clk'event and clk = '1' then
                iLAux <= iLAux_sat;
                VoutAux <= VoutAux_sat;
        end if;
    end process;

iLAux_redim <= resize (iLAux, iLAux_redim);
VoutAux_redim <= resize (VoutAux, VoutAux_redim);

iL <= to_slv (iLAux_redim);
Vout <= to_slv (VoutAux_redim);
```

Figura 15. Cálculo de las variables de estado. Modelo coma fija con librería *sfixed* del *full-bridge*.

```
process (dq, VgAux, VoutAux_redim, iLAdd)
```

```
begin
     case dq is
          when "01" =>
               iLAdd <= resize(-VgAux-VoutAux_redim, iLAdd);
          when "10" =>
               iLAdd <= VgAux - VoutAux_redim;
          when others =>
               iLAdd <= resize(-VoutAux_redim, iLAdd);
     end case;
end process;
```

Figura 16. Asignación de señales al multiplexor de corriente. Modelo coma fija con librería *sfixed* del *full-bridge.*

De la misma manera que con el modelo en coma flotante sintetizable, se estudia el problema de la resolución en coma fija. Como se vio con el modelo anterior, partiendo de los datos: i_L=2 A, I_R=1,8 A, v_{out}=200 V, se observa que la tensión de salida del siguiente ciclo de integración debe ser 200,000046 V. En este caso, se puede representar ese valor sin cometer ningún error ya la señal de tensión de salida tiene un formato de Q8.47, por lo que este valor es representable en dicho tamaño. Así, se observa la buena precisión que tiene este modelo. En este caso, el error por resolución es nulo ya que hay más bits para representar el valor deseado: 55 bits frente a 24. Aunque el modelo en sfixed utilice un número mayor de bits, se verá en el capítulo de Resultados que la frecuencia de síntesis será mucho mayor que en float32, mientras que los recursos utilizados son muy inferiores.

Otra ventaja del diseño en coma fija es que podemos optimizar cada señal con el ancho de palabra deseado. En el caso de las variables de estado, necesitamos representar el valor anterior de la señal de control (en el ejemplo 200 V), y a su vez el incremento ($4{,}6\cdot10^{-5}$ V), lo que sumaría 200,000046, y se guarda en 56 bits (formato Q8.47). Si no se utilizaran suficientes bits, no se integrarían con precisión los incrementos o incluso no podrían representarse. Sin embargo, a la hora de realimentar el modelo, no es necesario utilizar toda esa información, sino simplemente hacer una aproximación. En el ejemplo propuesto, se han utilizado un formato Q8.8 para la tensión de salida en la realimentación, por lo que la tensión de salida realimentada en el ejemplo sería 200 V.

La figura 17 muestra los tamaños de las señales del modelo en formato QX.Y.

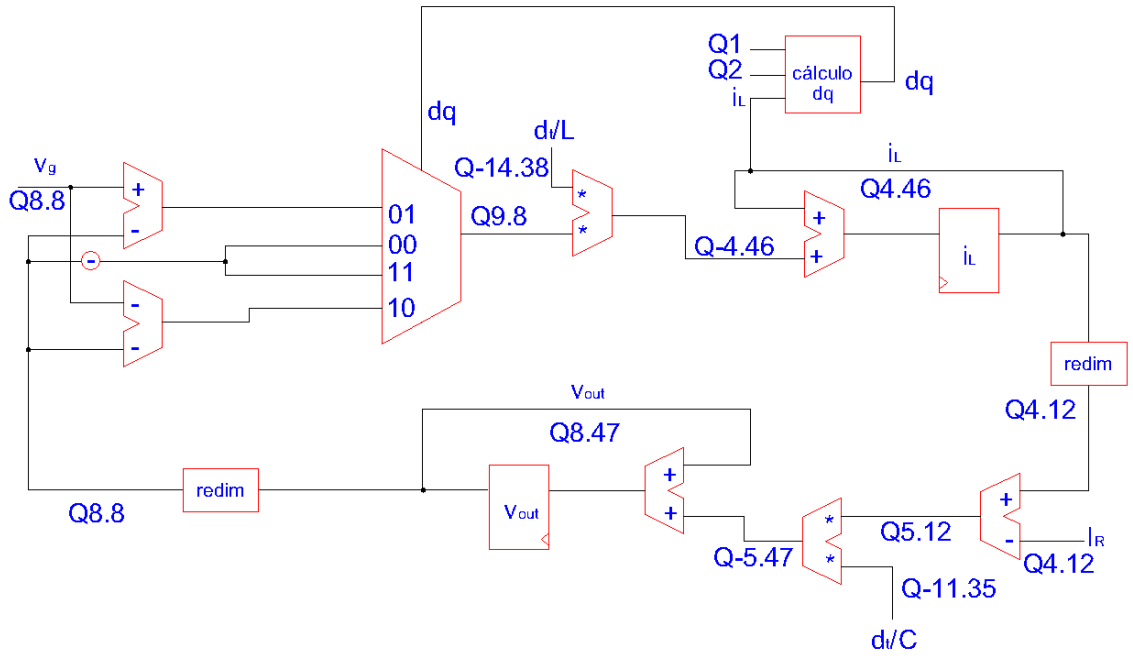


Figura 17. Esquema básico del *full-bridge* para el caso del modelado en formato QX.Y.

Se puede ver en la figura que tanto dtL como dtC tienen formatos del tipo Q-14.38. Esto significa que se representan valores decimales muy pequeños con -14 bits de parte entera y 38 de parte decimal, es decir, está representado desde el bit 15 decimal hasta el bit 38 decimal. Sin embargo, el número de bits de la señal se sigue calculando de la misma manera, 1 bit de signo, -14 bits de parte entera y 38 bits de parte decimal, es decir, 1 – 14 + 38 = 25 bits. Por ejemplo, siendo dt = $23 \cdot 10^{-9}$ seg y L = $0{,}9 \cdot 10^{-3}$ H, el valor del multiplicador es $2{,}55556 \cdot 10^{-4}$, siendo su valor en binario $0{,}00000000001000010111111100000101100000_2$, de donde se extrae el valor final $00010111111100000101100000_2$.

4.5 Modelo en coma fija parametrizable

El modelo en coma fija parametrizable, al igual que el anterior modelo, utiliza la notación QX.Y, aunque se trata de un modelo parametrizable, es decir, permitirá modificar los formatos QX.Y de todas las señales en tiempo de ejecución, sin necesidad de rediseñar el modelo. Para que esto sea posible, desde el exterior del modelo se definirá la escala de cada una de las entradas en función del tamaño y precisión necesarios. La escala, o cantidad de bits de la parte fraccionaria, se obtiene de manera que se resta al número total de bits, el menor entero mayor del logaritmo en base 2 del valor que se quiere representar. Por ejemplo, si la simulación debe soportar tensiones de entrada de hasta 20 V y estando dicha señal definida con 16 bits, la escala será $16 - \lceil \log_2 20 \rceil = 11$. Es decir, se utilizarían 5 bits para la parte entera junto al signo (con estos bits se puede llegar hasta el valor 31), quedando 11 bits de parte fraccionaria. Sin embargo, si debe soportar 1000 V, la escala será $16 - \lceil \log_2 1000 \rceil = 6$. En el primer caso la resolución será de 2^{-11} V, mientras que en el segundo caso será de 2^{-6} V, manteniendo una resolución aproximadamente proporcional al valor máximo soportado. Antes de realizar una simulación, pero sin necesidad de resintetizar el código, es necesario calcular las escalas de cada señal. Por tanto, para definir las escalas del modelo, se deben saber las condiciones con las que se va a operar el modelo, y realizar estos

cálculos. Estos cálculos pueden automatizarse a través de una aplicación para que la interfaz con el usuario sea fácil.

La figura 18 muestra el esquemático del full-bridge para el caso del modelado en formato coma fija parametrizable. Se pueden diferenciar dos partes: cálculo de la corriente de la bobina y cálculo de la tensión de salida. La primera parte consta de un multiplexor que, en función del estado de los transistores, elige la tensión aplicada en la bobina. Posteriormente se multiplica por $\frac{\Delta t}{L}$ y, finalmente, se acumula este incremento con la corriente del ciclo anterior. Este proceso se realiza de forma análoga para el cálculo de la tensión de salida.

En la figura 18 se pueden observar los tamaños de cada señal del modelo, pero su interpretación varía dependiendo de la escala de cada señal, también mostrada en la figura en forma de variables (V, dt/L, I, dt/C). Como se ha comentado, estas escalas dependen de los valores máximos esperados en las tensiones, tiempo de integración requerido (que depende de la frecuencia de conmutación), valor de la bobina, valor máximo esperado de la corriente y valor del condensador. El modelo se realimenta, así que no se pueden acumular las escalas, y es necesario realizar cambios de escala para poder operar correctamente con los datos de entrada, ya que, para sumar dos señales, estas tienen que tener la misma escala. Para este fin, la figura X muestra dos bloques de cambio de escala. Las entradas numéricas del modelo (Vg e Ir) deben estar representadas en las escalas requeridas, y las salidas deben interpretarse dependiendo de las correspondientes escalas, las cuales saldrán del modelo a través de DACs externos.

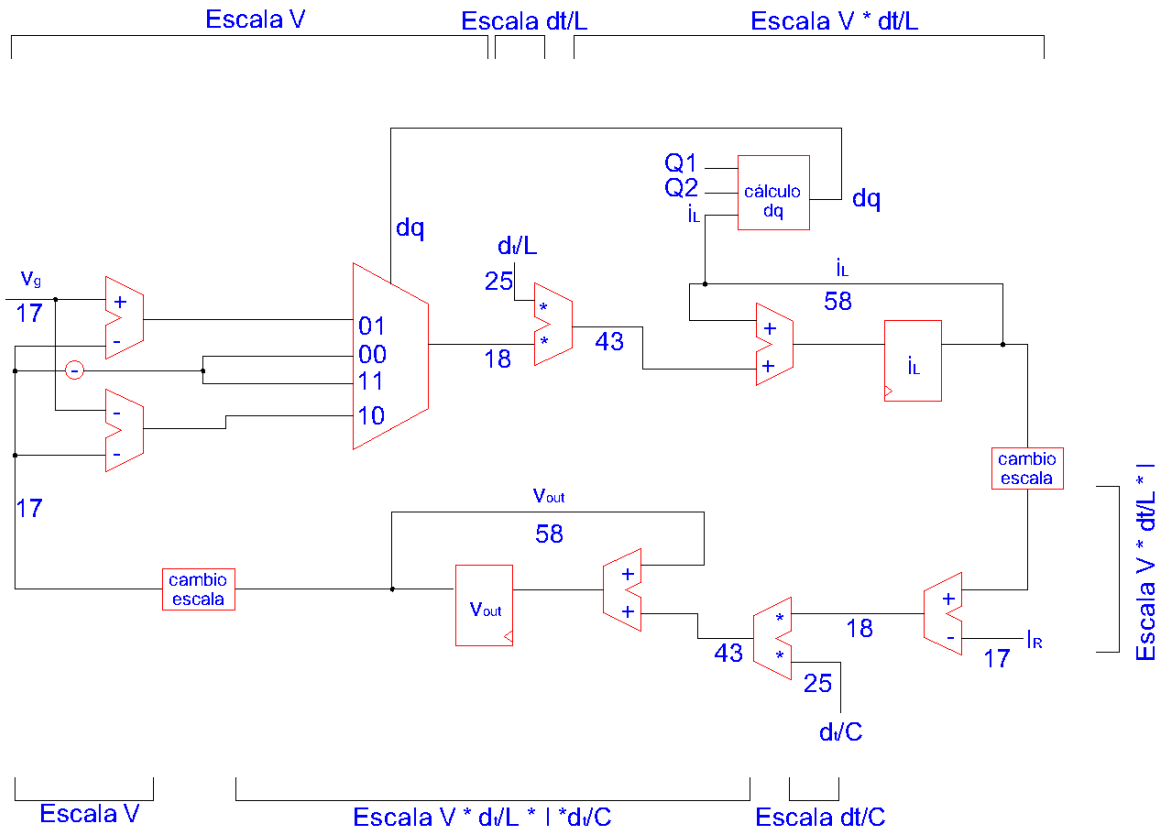


Figura 18. Implementación del modelo en coma fija parametrizable

Como se ha dicho anteriormente, el modelo estará implementado y puede configurarse sin necesidad de resintetizar el código. Por lo que, mediante una aplicación, se calcularán las escalas de cada señal y se configurarán los módulos de cambio de escala.

4.6 Integración con el sistema procesador

Como se comentó, entre los objetivos del TFG se encuentra la comunicación del modelo HIL con el exterior para que el modelo pueda ser configurado. Esta comunicación se realiza mediante una comunicación serie tipo UART. Aunque se puede implementar un módulo UART en VHDL o utilizando un IPCore, se ha optado por integrar el modelo HIL en un sistema basado en microprocesador ya que añadir una capa software al proyecto le otorga gran flexibilidad. Se ha utilizado una FPGA Zynq XC7Z020 de Xilinx, que consta de un procesador ARM con 2 cores. El sistema completo (figura 19) consistirá en el procesador y un bus AXI con tres periféricos: uno donde se integrará el modelo de la planta en coma fija parametrizable, otro con la UART y, por último, un periférico tipo Timer para gestionar las comunicaciones y algoritmia basada en tiempo.

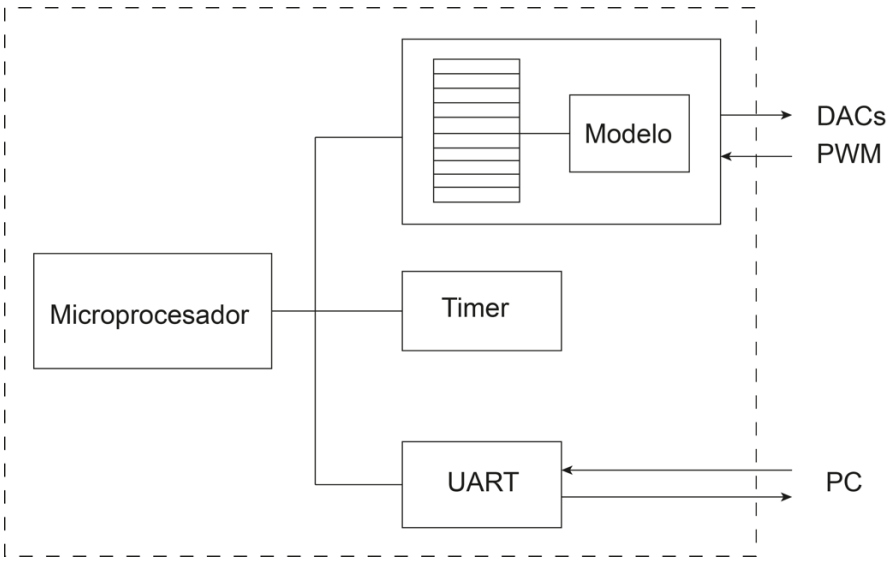


Figura 19. Sistema procesador completo.

Como se verá posteriormente, debido a la reducida cantidad de recursos hardware utilizada para el modelo HIL propuesto, se podrían integrar varios modelos como periféricos en paralelo, de tal manera que en una misma FPGA pudiera haber distintos modelos de convertidores conmutados, y el procesador se encargaría de la gestión y configuración de todos ellos.

4.6.1 Arquitectura

La comunicación entre el procesador y el periférico del modelo se realiza mediante la comunicación con registros. El modelo de la planta se integra en un periférico con 32 registros con 32 bits cada registro. Las señales tanto de entrada como de salida se asignan a cada registro escribiendo o leyendo de ellos los datos. Aunque el periférico se comunica con el procesador mediante los registros, el modelo también tiene las salidas *analógicas* que

representan las variables de estado del modelo, y que conforman la comunicación con el regulador del usuario. Estas salidas, todavía digitales, serán llevadas al exterior de la FPGA para ser conectadas a los DACs de alta velocidad que las conviertan en analógico. Además, el modelo necesita las entradas de control del usuario, que serán entradas a la FPGA.

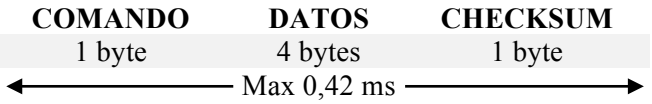
Se emplea un timer con dos canales para evitar problemas con las tramas enviadas y recibidas. Se utilizan dos timers porque tiene dos funciones. La primera es que se mandan periódicamente datos a la aplicación que hay en el ordenador dando información de las variables de estado, alarmas, etc. La segunda es que se implementan *watchdogs* cuando se recibe un byte de una trama. Esto se hace porque se supone la situación en la que ocasionalmente se pierde un byte de una trama. Si esto ocurre, se podría dar la desincronización entre aplicación y FPGA. Si envían 115200 bps, se envían 14400 Bps, por tanto, la diferencia temporal entre byte y byte es de 69,4 μs. Para que el mecanismo basado en *watchdog* funcione, el ordenador no mandará dos tramas seguidas con menos de 140 μs de separación (aproximadamente la separación entre dos bytes). Por lo que, si una vez que se ha recibido un byte de una trama no se recibe el siguiente en 140 μs, se descarta el buffer de recepción y se considerará que el próximo byte que llegue será nuevamente inicio de trama.

La configuración del modelo HIL se hace a través del puerto UART. Con un sistema de mensajes se pueden configurar todos los parámetros del modelo y sus alarmas. Además, cada segundo se envían los datos de las variables de estado y resultados de las alarmas, utilizando un canal del timer, como se dijo anteriormente.

4.6.2 Funcionamiento

La gestión de la recepción UART se realiza mediante interrupciones, en vez de usar *polling,* por lo que la carga del microprocesador disminuye sin perder prestaciones. Aunque el procesador no está actualmente saturado, se ha utilizado esta estrategia para que la escalabilidad del sistema sea mayor. De esta forma, en vez de estar preguntando continuamente al periférico UART si hay bytes recibidos, el procesador recibirá una interrupción cuando haya bytes recibidos, y podrá atenderla inmediatamente dejando de hacer lo que estuviera haciendo.

Cuando se recibe una interrupción, se comprueba el número de bytes recibidos y se inicia el *watchdog* para detectar tramas con pérdidas de paquetes. Si se reciben el número de bytes deseados antes que el *watchdog* se active, dichos bytes se considerarán como una trama completa. Las tramas completas se componen de 6 bytes: 1 byte con un comando para que el programa pueda interpretar qué configuración está siendo modificada, 4 bytes de mensaje que contiene el nuevo valor que se le asigna a dicha señal y otro byte de checksum para comprobar que la trama está bien formada y que la transmisión ha sido correcta.



La FPGA también calcula el checksum y, si coincide con el byte de checksum recibido, considerará la trama como válida. Se calcula haciendo la XOR de todos los datos anteriores que componen el mensaje.

$$\text{Checksum} = \text{CMD} \oplus \text{Dato}[0] \oplus \text{Dato}[1] \oplus \text{Dato}[2] \oplus \text{Dato}[3]$$

Por otro lado, el segundo canal del timer genera una interrupción cada segundo y dicha interrupción prepara una serie de tramas desde la FPGA a la aplicación del PC para informarle de las variables de estado y las alarmas. Estas tramas siguen el mismo protocolo en cuanto a formación de las tramas, tiempos máximos esperados, etc.

5. Resultados

Para comprobar la viabilidad del modelo parametrizable propuesto se deben realizar pruebas de precisión. La precisión del modelo viene determinada por tres factores principales: corrección en las fórmulas implementadas, elección del periodo de integración del modelo, y resolución en la aritmética usada.

Los dos primeros factores se pueden comprobar comparando el modelo de la planta junto a la función de transferencia extraída en el apartado 3.3. Dado que se quiere comprobar que las fórmulas extraídas para el modelo son correctas, se ha optado por comparar el modelo en coma flotante *real* con la función de transferencia del convertidor. Esta aritmética usada no presenta problemas de resolución ya que utiliza el estándar IEEE-754 de doble precisión, con 53 bits de mantisa, y su resolución es muy alta. Por tanto, funcionaría correctamente incluso para modelos con frecuencias de conmutación altas que requieren, por tanto, tiempo de integración muy bajo en los cálculos.

Una vez validadas las ecuaciones que modelan el convertidor, el último factor, es decir, la resolución en la aritmética usada, se puede comprobar comparando todos los modelos en VHDL.

Una vez se superan las pruebas de precisión, los modelos deben ser comparados en cuanto a cantidad de recursos hardware ocupados y frecuencia máxima de ejecución. Es importante destacar que la frecuencia máxima de ejecución determina el paso de integración ya que se desea que los modelos realicen emulaciones en tiempo real, es decir, el tiempo real del paso de integración coincide con el tiempo real que tarda en emularlo.

Algunos de los resultados que se van a ver en este capítulo van a ser publicados en un congreso nacional titulado “Seminario Anual de Automática, Electrónica Industrial e Instrumentación (SAAEI) 2016” y en un congreso internacional “Control and Modeling for Power Electronics (COMPEL) 2016” de la sociedad IEEE Power Electronics Society.

5.1 Precisión de la planta

Para la obtención de estos resultados de precisión, se ha elegido un tiempo de integración de 23 ns. Un bajo tiempo de integración ofrece resultados más precisos ya que se comete menos error en las ecuaciones en diferencias. En primer lugar, se ha comparado el modelo en coma flotante *real* en lazo abierto con la función de transferencia del convertidor calculada anteriormente. Los datos del modelo *real* se han extraído usando ModelSim 10.0b mientras que los datos de la función de transferencia se han extraído mediante Matlab R2011a.

Las pruebas de precisión deben hacerse en lazo abierto para comprobar si realmente el modelo está funcionando como debe. Si se añadiera un regulador en lazo cerrado, y el modelo cometiera una deriva en alguna de las variables de estado, el regulador corregiría dicho error y se podría pensar que el modelo está funcionando correctamente. En cambio, una deriva en lazo abierto no sería corregida y se detectaría fácilmente. En este apartado,

todas las comparaciones se han realizado sobre la tensión de salida regulada en lazo abierto, con distintas tensiones de entrada de 20 V y 200 V y con un ciclo de trabajo de d = 0,9.

La figura 20 muestra la superposición de la función de transferencia de un *full-bridge* con el resultado de la tensión de salida de modelo *real*.

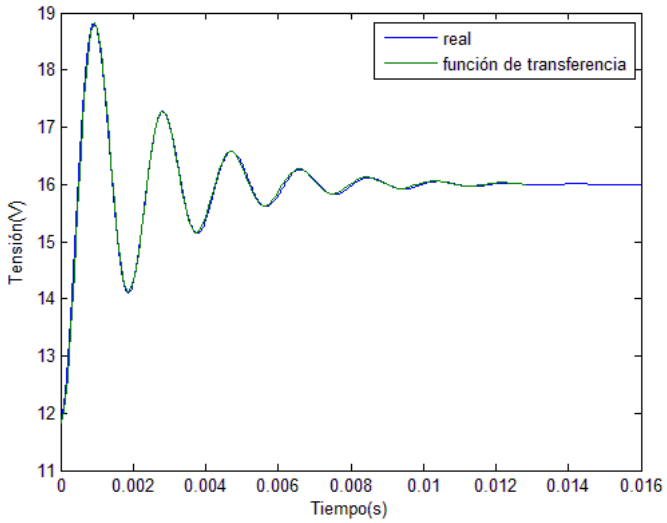


Figura 20. Comparación entre función de transferencia y modelo en *real* en lazo abierto.

Una vez se ha visto que las ecuaciones en diferencias son correctas, se van a probar el resto de aritméticas que sí son sintetizables, para comprobar si realmente pueden ser usadas en la emulación HIL propuesta. Se ha hecho una comparación del modelo *real* con el modelo *float32* primero con 20 V de tensión de entrada (figura 21) y después con 200 V de tensión de entrada (figura 22).

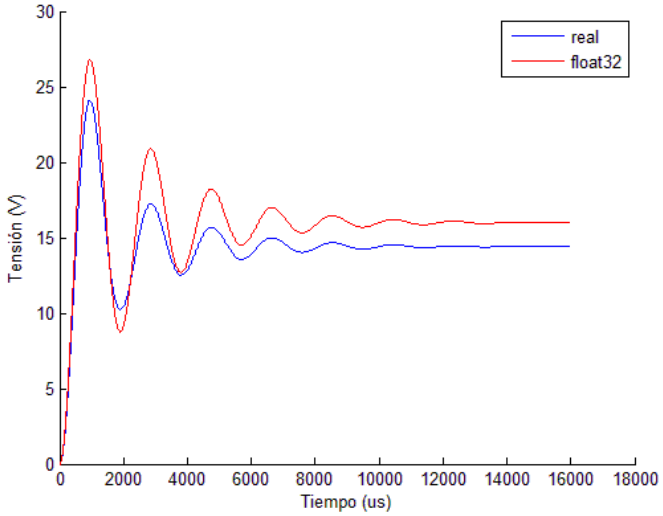


Figura 21. Comparación de tensiones de salida en lazo abierto con tensión de entrada de 20 V. Modelo en *real* comparado con modelo en *float32*.

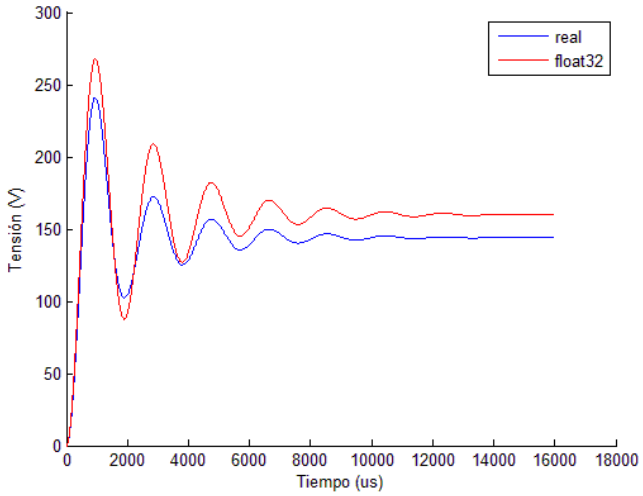


Figura 22. Comparación de tensiones de salida en lazo abierto con tensión de entrada de 200 V. Modelo en *real* comparado con modelo en *float32*.

Se comprueba la baja precisión de este modelo calculando sus respectivos errores absolutos (figura 23 y figura 24). Teniendo la primera un error absoluto medio de 1,5906 V con una desviación típica de 0,5901 V y la segunda un error absoluto medio de 15,9056 V con una desviación típica de 5,9009 V. Como se ha explicado anteriormente, el origen de este problema es la falta de resolución de la coma flotante en 32 bits cuando el tiempo de integración es bajo y, por tanto, los incrementos de tensión y corriente son pequeños.

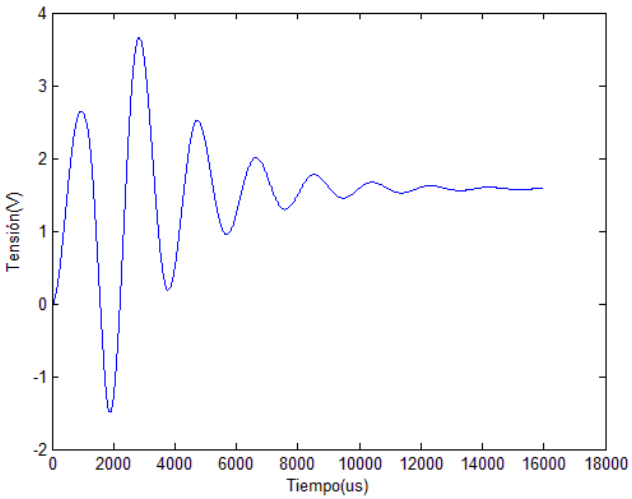


Figura 23. Error absoluto de las tensiones de salida en lazo abierto con tensión de entrada de 20 V. Modelo en *real* comparado con modelo en *float32*.

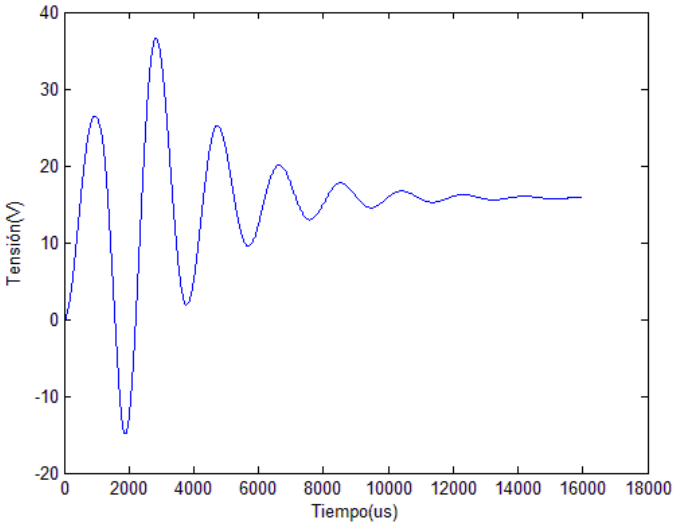


Figura 24. Error absoluto de las tensiones de salida en lazo abierto con tensión de entrada de 200 V. Modelo en *real* comparado con modelo en *float32*.

La figura 25 y la figura 26 muestran la comparación del modelo *real* ahora con el modelo en coma fija con librería *sfixed* con 20 V y 200 V de tensión de entrada respectivamente. En coma fija, como se dijo anteriormente, se requiere elegir cuántos bits hay para la parte entera y cuantos hay para la parte decimal. Como se han elegido los bits para el caso de tensión de entrada igual a 20 V, todo el modelo está optimizado y se obtienen precisiones similares con menos bits.

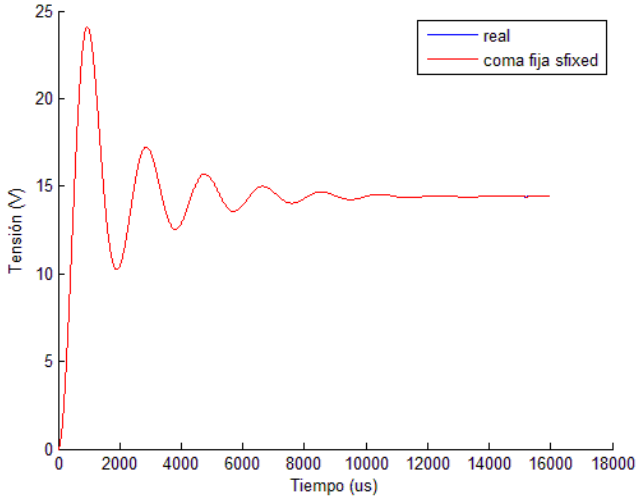


Figura 25. Comparación de tensiones de salida en lazo abierto con tensión de entrada de 20 V. Modelo en *real* comparado con modelo en coma fija con librería *sfixed*.

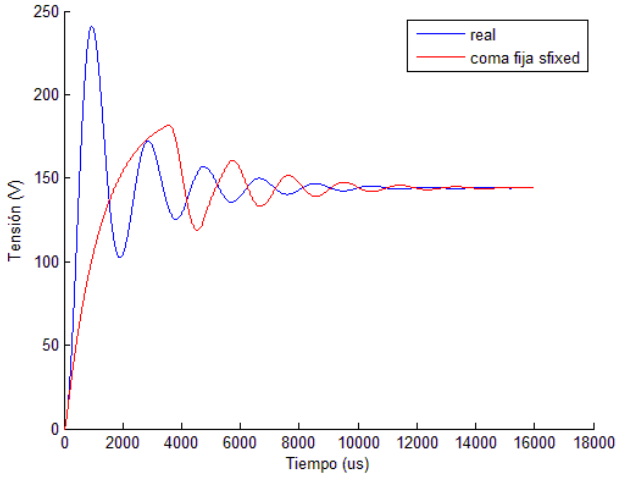


Figura 26. Comparación de tensiones de salida en lazo abierto con tensión de entrada de 200 V. Modelo en *real* comparado con modelo en coma fija con librería *sfixed.*

Los tamaños de las señales se fijan antes de implementar el modelo, por lo que, una vez fijado el tamaño, aun cambiando el valor de los datos de entrada, estos no varían. Es por eso por lo que para el caso en que la tensión de entrada son 200 V, teniendo en cuenta que los tamaños se habían prefijado para un valor de entrada de 20 V, el número de bits de la parte entera no es suficiente como para soportar un valor tan elevado, por lo que este modelo satura, tal y como se puede observar. Suponiendo un número total de bits fijos, lo óptimo hubiera sido asignar más bits para la parte entera sacrificando resolución. Sin embargo, en *sfixed* este cambio de bits requiere resintetizar y, por tanto, cambiar el modelo en cada ejecución.

Se calcula el error absoluto tanto de la figura 24 como de la figura 25, estando estos representados en la figura 27 y en la figura 28, respectivamente.

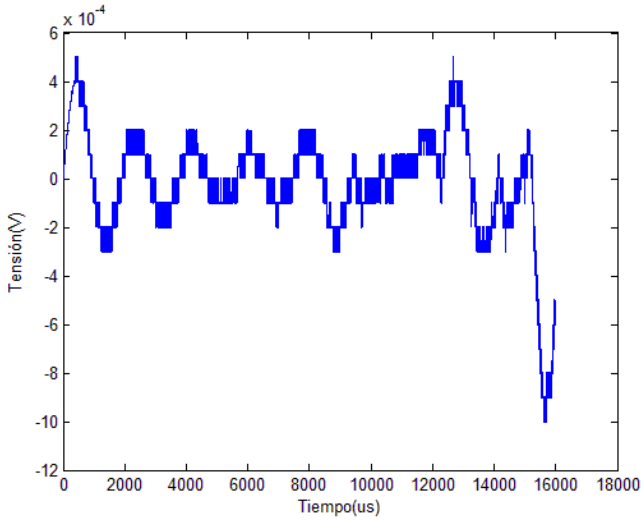


Figura 27. Error absoluto de las tensiones de salida en lazo abierto. Modelo en *real* comparado con modelo en coma fija con librería *sfixed* con 20 V de tensión de entrada.

Se observa que el error disminuye a medida que la tensión se va acercando al régimen permanente. Esto es porque los transitorios son los puntos donde la falta de resolución penaliza a la precisión del modelo. Para las pruebas de tensión de entrada igual a 20 V, la

tensión de salida tiene un error absoluto medio de 0,00014369 V con una desviación típica de 0,00016365 V. El error se ha medido en valor absoluto, porque un error positivo y otro error posterior negativo no deben contrarrestarse, sino que ambos son errores que deben ser contabilizados para calcular el error global del modelo.

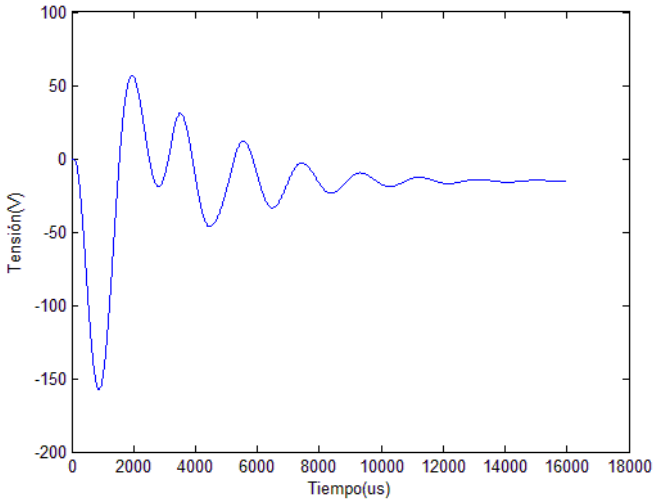


Figura 28. Error absoluto de las tensiones de salida en lazo abierto. Modelo en *real* comparado con modelo en coma fija con librería *sfixed* con 200 V de tensión de entrada.

Como era de esperar, la figura 28 (tensión de entrada 200 V) tiene un error muy elevado. La tensión de salida tiene un error absoluto medio de 15,9983 V con una desviación típica de 27,6699 V. El error tan elevado es debido a que alguna señales del modelo llegan a sus valores máximos de representación y se saturan. Aunque se ha evitado que haya desbordamientos aritméticos, la saturación provoca errores notables.

Para comprobar la versatilidad del modelo en coma fija parametrizable, al igual que en el caso anterior, se ha comparado dicho modelo con el modelo en coma flotante *real* para los dos casos de tensión de entrada, 20 V y 200 V (figura 29 y figura 30).

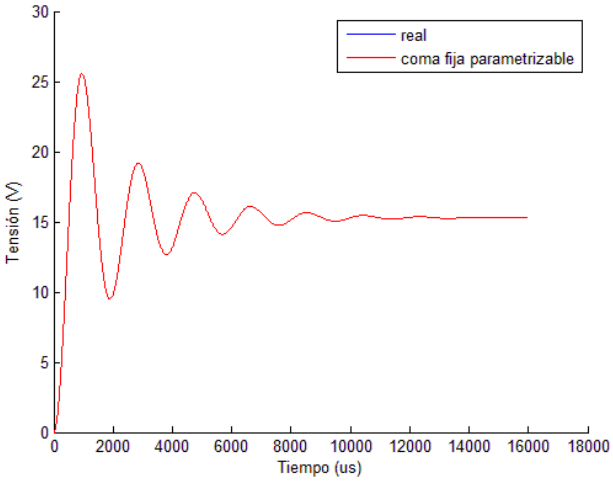


Figura 29. Comparación de tensiones de salida en lazo abierto con tensión de entrada de 20 V. Modelo en *real* comparado con modelo en coma fija parametrizable.

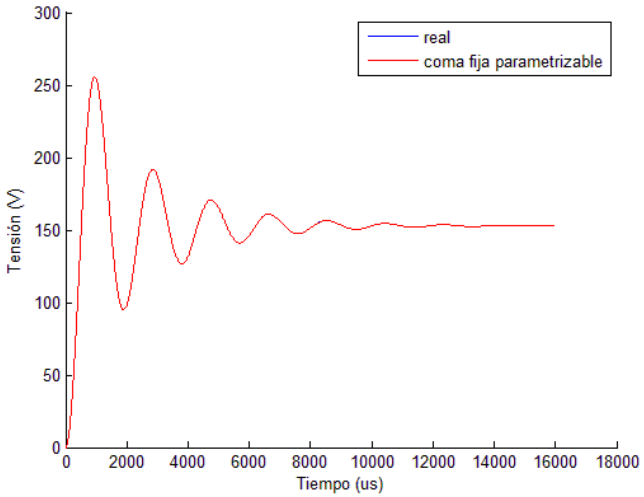


Figura 30. Comparación de tensiones de salida en lazo abierto con tensión de entrada de 200 V. Modelo en *real* comparado con modelo en coma fija parametrizable.

Para comparar la precisión de este modelo, se calcula el error, al igual que se ha hecho anteriormente con el resto de los modelos, representados en la figura 31 y en la figura 32. En el primer caso, la tensión de salida tiene un error absoluto medio de 0,00012911 V y una desviación típica de 0,000090655 V, mientras que en el segundo caso el error absoluto medio es de 0,0011 V y la desviación típica de 0,00082317 V.

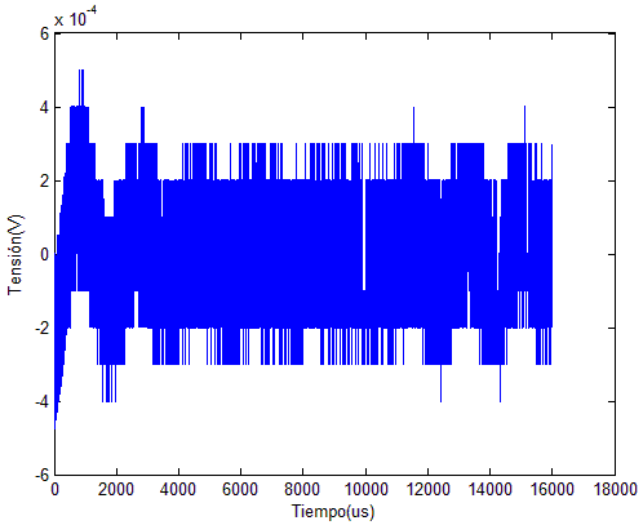


Figura 31. Error absoluto de las tensiones de salida en lazo abierto con tensión de entrada de 20 V. Modelo en *real* comparado con modelo en coma fija parametrizable.

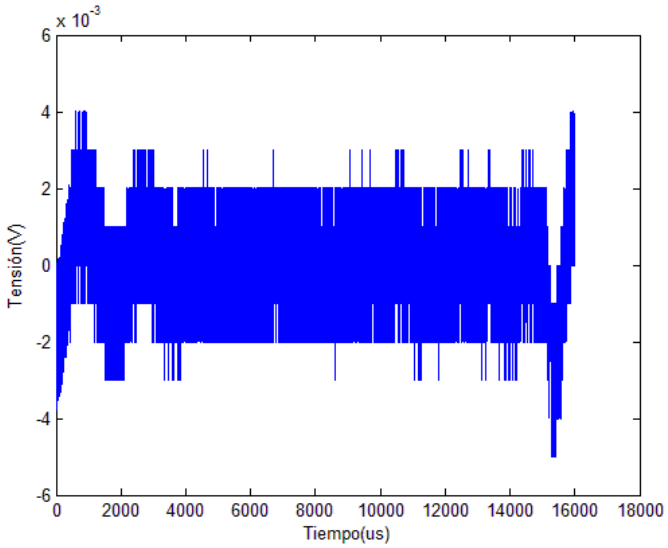


Figura 32. Error absoluto de las tensiones de salida en lazo abierto con tensión de entrada de 200 V. Modelo en *real* comparado con modelo en coma fija parametrizable.

Dado que el modelo en coma fija parametrizable puede configurarse sin necesidad de resintetizar, se pueden adaptar las escalas a las condiciones de simulación, consiguiendo una precisión similar en términos relativos. Así, multiplicando por 10 la tensión de entrada, se ha obtenido un error aproximadamente 10 veces mayor. Este aumento de error, a efectos prácticos, no es crítico ya que un error en torno al 0,0013% (error de 0,0001985 V frente a una tensión de salida de 15 V) suele ser asumible.

Por tanto, se comprueba que el modelo en coma fija parametrizable tiene una precisión mayor que la coma flotante sintetizable de 32 bits, y que el modelo de coma fija *sfixed* ante condiciones de simulación diferentes a las nominales. La tabla I resume la precisión de cada uno de los modelos con respecto al modelo *real*, el modelo de referencia.

TABLA I
COMPARATIVA DE ERRORES Y DESVIACIONES TÍPICAS DE LOS MODELOS

Sistema	**Error absoluto (V)**		**Desviación típica (V)**	
	Vg = 20 V	**Vg = 200 V**	**Vg = 20 V**	**Vg = 200 V**
Tipo float32	1,5906	15,9056	$5{,}901 \cdot 10^{-1}$	5,9009
Coma fija *sfixed*	$1{,}4369 \cdot 10^{-4}$	15,9983	$1{,}6365 \cdot 10^{-4}$	27,6699
Coma fija parametrizable	$1{,}2911 \cdot 10^{-4}$	$1{,}1 \cdot 10^{-3}$	$9{,}0655 \cdot 10^{-5}$	$8{,}2317 \cdot 10^{-4}$

Excepto el modelo en coma fija con librería *sfixed*, en el cual el error es enorme debido al tamaño de las señales mencionado anteriormente, como la tensión de entrada ha aumentado en 10 aproximadamente, tanto el error como la desviación típica también han aumentado en 10.

5.2 Sistema HIL

Los datos de la sección anterior, han sido extraídos con un simulador VHDL para probarlo, sin embargo, el sistema final extrae su información a través de los DACs. Para ello, el modelo en coma fija parametrizable se implementa en una FPGA Zynq XC7Z020, configurando el modelo a través de la aplicación de prueba mostrada en la figura 33.

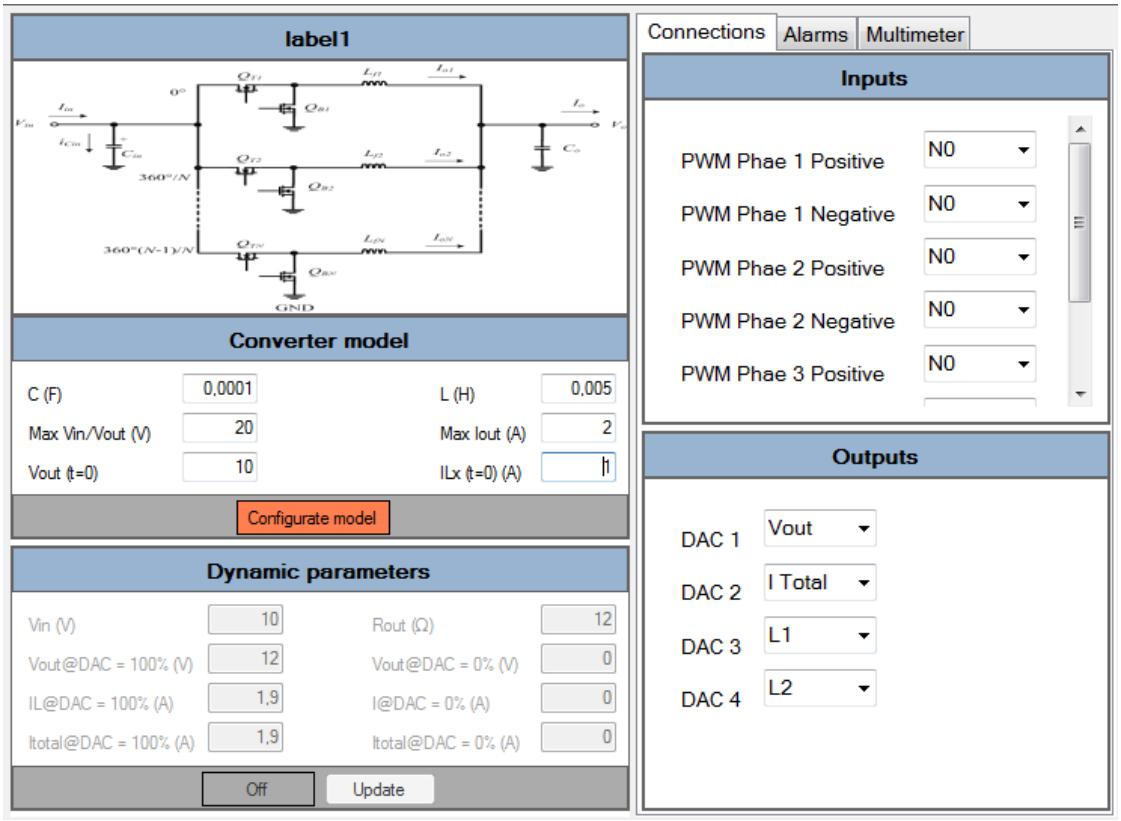


Figura 33. Aplicación para configuración del modelo en coma fija parametrizable.

La figura 34 muestra la integración del modelo HIL en un sistema basado en microprocesador para su implementación en la FPGA mediante la herramienta Vivado 2015.4.

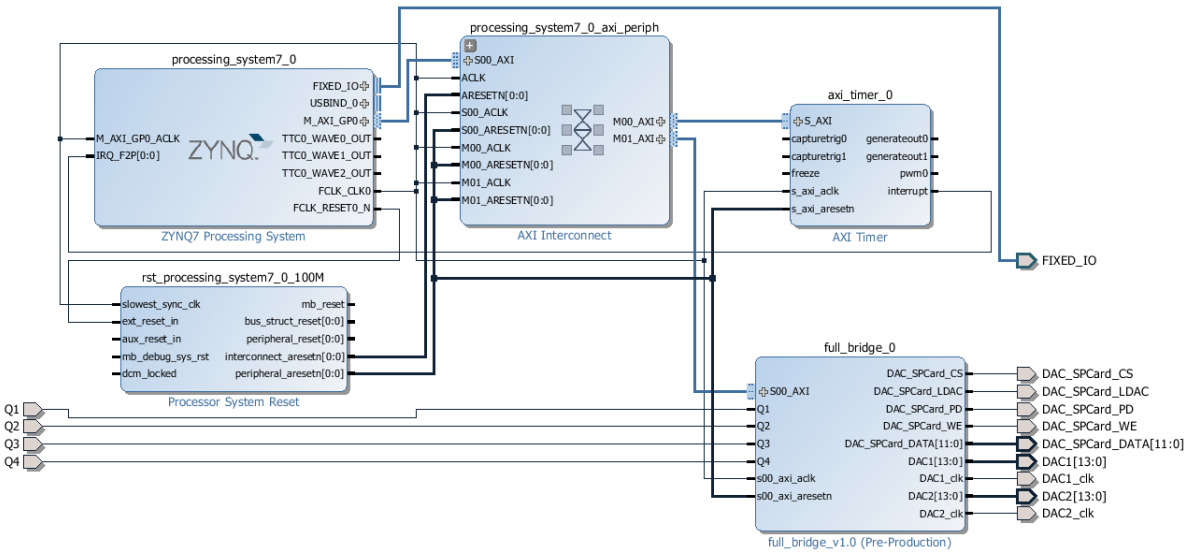


Figura 34. Integración del modelo HIL en un sistema basado en microprocesador.

Tal y como se ha dicho en el apartado 4.6, la FPGA contiene unos DACs externos por los que salen las variables de estado: la tensión de salida, la corriente de la bobina y la corriente de entrada. Las salidas de estos DACs se conectan a sondas de un osciloscopio MSO-X 3104A, siendo estas salidas las que se conectarían a un hipotético control en lazo cerrado, obteniendo la tensión de salida (azul), la tensión de la bobina (amarilla) y la tensión de entrada (verde) de la figura 35. Por otra parte, el ciclo PWM es generado exteriormente desde otra FPGA Zynq XC7Z020 conectada al modelo. Esta segunda FPGA está haciendo de regulador en lazo abierto. Como se comentaba en la Introducción, gracias a que el sistema HIL es un sistema con entradas (señales de control) y salidas analógicas, cualquier otro tipo de control puede ser probado junto al sistema HIL, siempre que su entrada sea de tensión.

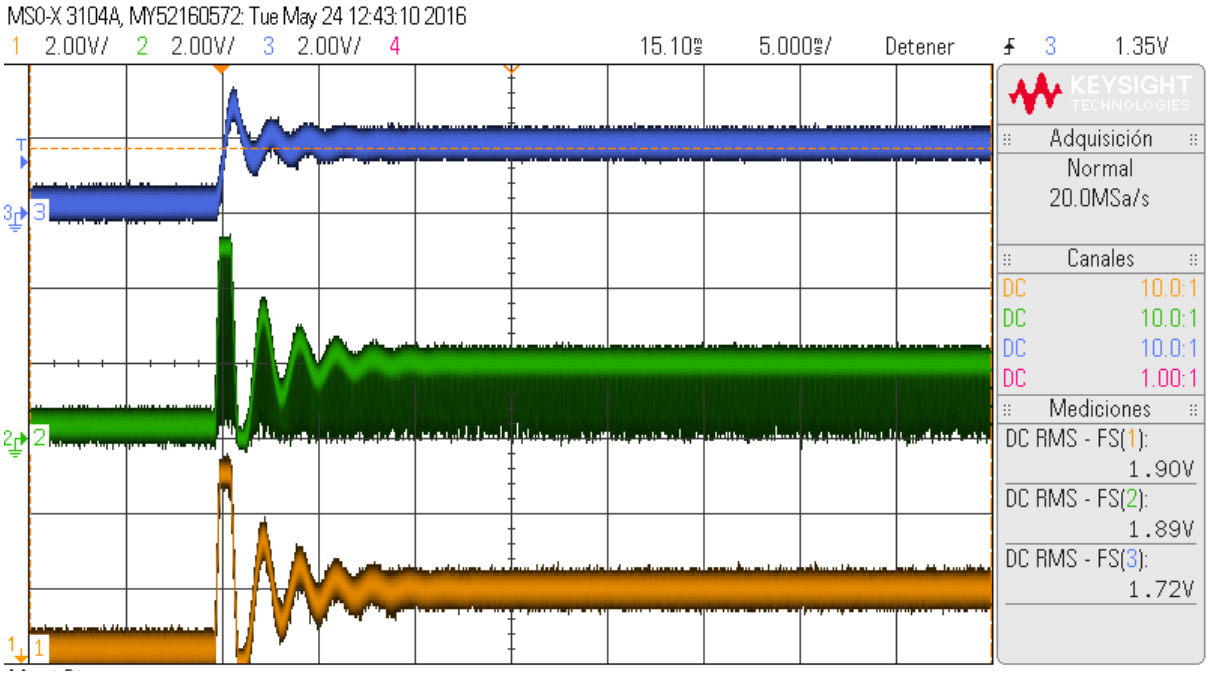


Figura 35. Tensión de salida del modelo HIL.

Tal y como se puede ver en la figura L, la corriente de entrada es muy similar a la corriente de la bobina, aunque se ve un alto rizado entre 0 V y el valor de la corriente de la bobina.

Esto se debe a que la corriente de entrada puede ser igual a la corriente de la bobina cuando los interruptores tienen alguna de las combinaciones dq = "01" o dq = "10", según la notación de la figura 11. Según la figura 36, la corriente de entrada también puede ser igual a la corriente de la bobina negativa, sin embargo, el DAC no representa corrientes negativas. En dicho caso, el DAC da un valor de 0.

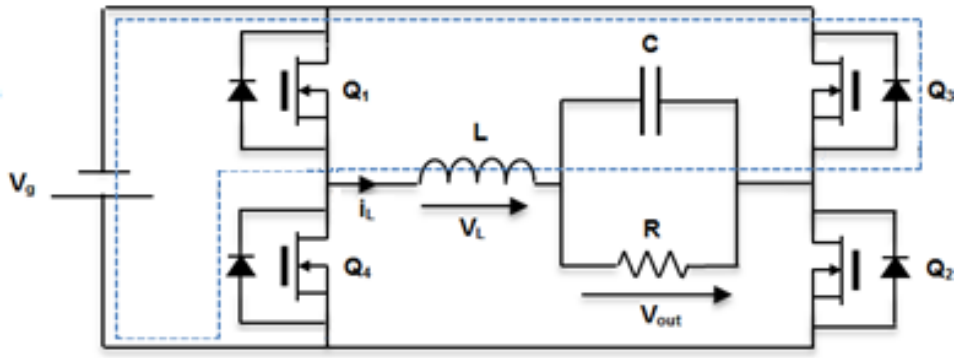


Figura 36. Corriente de un convertidor inversor en tiempo de *deadtime*.

El usuario puede elegir las tensiones/corrientes que se representan cuando el DAC esté dando el 0% y el 100% de su salida. Estos valores se darán en función de lo que el control quiera leer. Como los valores no tienen por qué ser potencias de 2, no es suficiente con un simple desplazador. El modelo debe quitar la escala interna de la variable de control y debe ajustar la salida a las escalas que quiere el usuario. Por lo que a los valores de 17 bits se les resta el offset y se multiplica por la escala del DAC, calculada a partir de la tensión/corriente máxima del DAC, del offset y de la escala de tensión de entrada o de la corriente.

En la placa de pruebas, se han utilizado 3 DACs, 2 de alta velocidad (DAC904U) [18] y otro de menos velocidad integrado en los DAC (TLV5619IPW), Los DACs, de alta velocidad suelen tener su salida en corriente, por lo que se conecta un amplificador operacional CLC1005 para obtener una salida modulada en tensión.

Aunque el sistema HIL permite elegir qué dato saldrá por cada DAC, es recomendable usar los DACs de alta velocidad para extraer los datos equivalentes a las corrientes de la bobina y de entrada, ya que las corrientes tienen mayor rizado que la tensión de salida.

Otra diferencia entre los DACs de alta velocidad y de baja velocidad es que los primeros tienen una salida de 5 V gracias al amplificador operacional, mientras que el segundo tiene una salida máxima de 3,3 V.

5.3 Área y velocidad de la planta

En este apartado se van a mostrar la velocidad de las simulaciones y emulaciones de los modelos propuestos. El tiempo de simulación es útil si se quisieran hacer pruebas del modelo junto a un regulador VHDL. Sin embargo, los resultados principales son los de emulación, los cuales permitirían separar el control y el modelo, y utilizar la arquitectura HIL completa con DACs. Por otra parte, los resultados de emulación son enormemente mejores en comparación a los resultados de simulación, como se podrá comprobar.

El tiempo de emulación es uno de los criterios de comparación fundamental, porque determinará cuál es el mínimo periodo de integración que se puede obtener. En el apartado 5.1 se mostraron los resultados de precisión de diferentes modelos. Sin embargo, estos resultados sólo tienen sentido si dichos modelos pueden implementarse a la frecuencia de reloj con los que han sido simulados. Es decir, si el periodo de reloj resultante de la implementación es mayor que el tiempo de integración usado en el apartado 5.1, la emulación no se ejecutaría en tiempo real. Para conseguir tiempo real, debería bajarse el tiempo de integración y el error de las ecuaciones en diferencias aumentaría.

Aunque la motivación de este trabajo es emular (implementar), también se van a dar datos de simulación ya que demuestra qué modelos son más costosos. Para calcular los tiempos de simulación se ha probado a realizar una simulación de 100 ms del convertidor. La simulación se realizó en un ordenador con procesador Intel Core i7-4770 a 3,4 GHz y 8 GB de RAM. La tabla II recoge dichos resultados.

TABLA II
RESULTADOS DE TIEMPO DE SIMULACIÓN DE 100 MS

Sistema	Simulación/Emulación	Tiempo
Tipo real	Simulación	54” 120 ms
Tipo float32	Simulación	6’ 55” 18 ms
Coma fija sfixed	Simulación	2’ 22” 89 ms
Coma fija parametrizable	Simulación	1’ 27” 70 ms
Coma fija sfixed	Emulación	100 ms
Coma fija parametrizable	Emulación	100 ms

Se observa que el tipo *real* es rápido, pero sólo porque utiliza coma flotante utilizando las instrucciones nativas del procesador del ordenador. El tipo *float32* es muy lento porque simula el hardware de verdad que tendrá un modelo *float32* y dicha aritmética es muy compleja. Y, por último, los tipos en coma fija con librería *sfixed* y parametrizable son muy parecidos y mucho más rápidos porque son sencillos. Su parecido se debe a que la única diferencia considerable son los módulos de cambio de escala.

Por otro lado, la tabla III muestra los recursos utilizados por la FPGA para implementar el sistema HIL.

TABLA III
RECURSOS USADOS EN EL DISEÑO EN UNA FPGA XILINX XC7Z020-1

Sistema	Máx. Frec.	LUTs		FFs		Mult	
		Usado	%	Usado	%	Usado	%
Coma flotante tipo float32	12,20 MHz	8987	16	69	1	4	1
Modelo Coma fija librería sfixed	46,84 MHz	415	1	109	0,1	3	1
Modelo Coma fija parametrizable	45,22 MHz	509	1,5	106	0,1	3	1

Como se puede observar, se utilizan una cantidad muy reducida de recursos, por lo que se podrían implementar varios modelos en paralelo, verificando lo dicho anteriormente en el apartado 4.6, se podrían crear modelos con pérdidas eléctricas o se podría utilizar una FPGA de coste menor.

Se comprueba que, poniendo un tiempo de integración de 23 ns, la frecuencia de implementación en coma fija y en parametrizable es alrededor de los 45 MHz, pudiendo poner así un reloj de 23 ns y así ir en tiempo real. En coma flotante tipo *float32*, no se iría en tiempo real, además, se había visto que tiene un precisión mala.

6. Conclusiones

6.1 Conclusiones

La importancia de la simulación de un regulador antes de ser probado en un convertidor de potencia está fuera de duda. No sólo es necesario diseñar el regulador realizando simulaciones de comportamiento junto a un modelo de la planta, sino que también es recomendable realizar pruebas del regulador cuando ya está implementado. Sin embargo, no hay una técnica estándar para este tipo de simulaciones. La simulación final recomendable sería poder probar el control en su estado final, implementado en un microprocesador, FPGA, DSP, etc., y conectar el sistema de control a un dispositivo que implemente un modelo de la planta, utilizando la técnica HIL. En este Trabajo Fin de Grado se han planteado distintos métodos de implementación para el modelado HIL de un convertidor *full-bridge* en HDL: coma flotante, coma fija y coma fija parametrizable.

El modelo en coma flotante no sintetizable (con señales tipo *real*) es muy sencillo de implementar y está soportado por la mayoría de los simuladores, sin embargo, no es sintetizable, por lo que no es válido para los objetivos del TFG. El modelo en coma flotante sintetizable no es viable para convertidores conmutados debido a su falta de resolución cuando el tiempo de integración es muy bajo, y dicho tiempo debe ser bajo cuando la frecuencia de conmutación es alta (centenas de hertzios). La alternativa en coma fija con librería *sfixed* permite obtener frecuencias de síntesis más altas utilizando menos recursos. Sin embargo, el ancho de cada variable del modelo debe ser ajustado de forma personalizada. Además, el ancho se debe escoger teniendo en cuenta las condiciones previstas de simulación, por lo que un cambio en los datos de la simulación puede hacer que el modelo no funcione correctamente debido a que no tenga suficiente resolución con los nuevos valores o, en cambio, si haya resolución pero se saturen las variables por alcanzar valores demasiado grandes. Por último, en el Trabajo Fin de Grado se ha presentado un modelo en coma fija parametrizable, que posee las propiedades de la coma fija (alta velocidad de síntesis y bajos recursos ocupados) pero que, a la vez, se puede adaptar a cualquier rango de valores, ya que el lugar de la coma es configurable sin necesidad de resintetizar el modelo.

El TFG muestra una comparativa de todos los modelos, en los que se han valorado criterios como precisión, frecuencia de síntesis y área. Primero se ha comprobado que el modelo *real*, que implementa coma flotante de doble precisión pero no simulable, modela correctamente la planta deseada, por lo que se ha considerado como el punto de referencia para todas las pruebas de precisión. Las pruebas demuestran que el modelo en coma flotante no tiene suficiente precisión, mientras que el error en los modelos en coma fija es mucho menor. Además, cuando las condiciones de simulación cambian, el modelo en coma fija parametrizable puede adaptarse por lo que el error porcentual se mantiene, salvando la gran desventaja de la coma fija clásica.

Con los resultados se puede concluir que el modelo en coma fija parametrizable es un modelo de gran precisión (con errores alrededor de microvoltios cuando la tensión de salida está en el orden de las decenas de voltios). Además, se ha comprobado que la emulación en FPGA de este modelo permite verificaciones rápidas, llegando a tiempo real si el periodo de integración está en el orden de las decenas de nanosegundos.

Como resultado de este TFG se van a presentar dos artículos titulados “Emulación de convertidores conmutados utilizando modelos en coma fija parametrizable” y “Hardware-in-the-loop using parametrizable fixed point notation” en el congreso nacional Seminario Anual de Automática, Electrónica Industrial e Instrumentación 2016 (SAAEI) y en el congreso internacional IEEE Workshop on Control and Modeling for Power Electronics 2016 (COMPEL)

6.2 Trabajo futuro

De los resultados obtenidos en este Trabajo Fin de Grado surgen nuevas líneas de investigación y mejora. Las líneas de investigación futura y posibles mejoras propuestas sobre el trabajo realizado son las siguientes:

- Aunque las pérdidas del modelo no son totalmente importantes para HIL, porque lo que se pretende es probar la implementación final del regulador, si se quiere un modelo 100% realista, hay que tener en cuenta las pérdidas eléctricas de primer orden. Estas pérdidas serían las generadas por:
    - La resistencia en serie de la bobina.
    - La resistencia en conducción del MOSFET.
    - La tensión del codo del diodo.
    - La resistencia en serie del condensador.
- Tener la posibilidad de unir, a través de la aplicación de prueba, dos topologías en serie. Por ejemplo, es típico hacer PFC (*Power Factor Correction*) con un boost y un buck en serie, por lo que, sería deseable unir las salidas de las variables de estado de un modelo a las entradas del siguiente.

Referencias

[1] A. De Castro. Aplicación del Control Digital basado en Hardware Específico para Convertidores de Potencia Conmutados. *Tesis Doctoral*. 2003.

[2] A. Prodic and D. Maksimovic, "Mixed-signal simulation of digitally controlled switching converters," in Proc. IEEE Workshop Comput. Power Electron., Mayaguez, PR, USA, Jun. 2002, pp. 100–105.

[3] L. Barragan, I. Urriza, D. Navarro, J. Artigas, J. Acero, and J. Burdio, "Comparing simulation alternatives of FPGA-based controllers for switching converters," in Proc. IEEE Int. Symp. Ind. Electron. (ISIE), Boulder, CO, USA, Jun. 2007, pp. 419–424.

[4] I. Urriza, L. Barragan, J. Artigas, J. Acero, D. Navarro, and J. Burdio, "Using mixed-signal simulation to design a digital power measurement system for induction heating home appliances," in Industrial Electronics, 2007. ISIE 2007. IEEE International Symposium on, pp. 1447-1451, jun. 2007.

[5] http://www.cadence.com/rl/Resources/datasheets/virtuoso_ade_fam_ds.pdf Última revisión: 25 de mayo de 2016.

[6] https://www.mentor.com/products/ic_nanometer_design/analog-mixed-signal-verification/advance-ms/ Última revisión: 25 de mayo de 2016.

[7] http://www.gecko-simulations.com/geckocircuits.html Última revisión: 25 de mayo de 2016.

[8] P. Zumel, M. García-Valderas, A. Lázaro, C. López-Ongil, and A. Barrado, "Co-simulation PSIM-ModelSim oriented to digitally controlled switching power converters," in Proc. IEEE 12th Workshop Control Model. Power Electron. (COMPEL), Jun. 2010, pp. 1–7.

[9] S. Karimi, P. Poure, and S. Saadate, "An HIL-based reconfigurable platform for design, implementation, and verification of electrical system digital controllers," IEEE Trans. Ind. Electron., vol. 57, no. 4, pp. 1226–1236, Apr. 2010.

[10] M. Matar and R. Iravani, "FPGA implementation of the power electronic converter model for real-time simulation of electromagnetic transients," IEEE Trans. Power Del., vol. 25, no. 2, pp. 852–860, Apr. 2010.

[11] Y. Chen and V. Dinavahi, "Digital hardware emulation of universal machine and universal line models for real-time electromagnetic transient simulation," IEEE Trans. Ind. Electron., vol. 59, no. 2, pp. 1300–1309, Feb. 2012.

[12] C. Dufour, V. Lapointe, J. Belanger, and S. Abourida, "Hardware-in-the-loop closed-loop experiments with an fpga-based permanent magnet synchronous motor drive system and a rapidly prototyped controller," in Industrial Electronics, 2008. ISIE 2008. IEEE International Symposium on, pp. 2152-2158, jul. 2008.

[13] M. Matar and R. Iravani, "Fpga implementation of the power electronic converter model for real-time simulation of electromagnetic transients," Power Delivery, IEEE Transactions on, vol. 25, pp. 852-860, apr. 2010.

[14] O. Lucia, I. Urriza, L. Barragan, D. Navarro, O. Jimenez, and J. Burdio, "Real-time FPGA-based hardware-in-the-loop simulation test bench applied to multiple-output power converters," IEEE Trans. Ind. Appl., vol. 47, no. 2, pp. 853–860, Mar.–Apr. 2011.

[15] O. Lucia, O. Jiménez, L. Barragán, I. Urriza, J. Burdío, and D. Navarro, "Realtime fpga-based hardware-in-the-loop development test-bench for multiple output power converters," in Applied Power Electronics Conference and Exposition (APEC), 2010 Twenty-Fifth Annual IEEE, pp. 309-314, feb. 2010.

[16] A. Sanchez, A. de Castro & J. Garrido, "A comparison of simulation and hardware-in-the-loop alternatives for digital control of power converters", in IEEE Transactions on Industrial Informatics, vol. 8, no. 3, pp. 491-500, ago 2012.
[17] Erickson R., W. y Maksimovic D. (2004). Fundamentals of Power Electronics, Kluwer Academic Publisher. Second Edition. New York.
[18] Javier Casatorres Agüero, "Diseño e implementación de módulos para la emulación de fuentes de alimentación en lazo cerrado". Trabajo Fin de Grado. 2015.
[19] I. Villar, A. Sanchez, A. de Castro, F. López-Colino and J. Garrido, "Emulación de convertidores conmutados utilizando modelos en coma fija parametrizable," in Seminario Anual de Automática, Electrónica Industrial e Instrumentación (SAAEI), 2016. En prensa.
[20] A. Sanchez, I. Villar, A. de Castro, F. López-Colino and J. Garrido, "Hardware-in-the-loop using parametrizable fixed point notation," in IEEE Workshop on Control and Modeling for Power Electronics (COMPEL), 2016. En prensa.

Glosario

AXI	*Advance eXtensible Interface*
CCM	*Continuous Conduction Mode*
DAC	*Digital to Analog Converter*
DCM	*Discontinuous Conduction Mode*
DSP	*Digital Signal Processor*
FPGA	*Field-Programmable Gate Array*
HDL	*Hardware Description Language*
HIL	*Hardware In-the-Loop*
MOSFET	*Metal-oxide-semiconductor Field-effect transistor*
PFC	*Power Factor Correction*
PWM	*Pulse Width Modulation*
UART	*Universal Asynchronous Receiver-Transmitter*

Anexos

Emulación de convertidores conmutados utilizando modelos en coma fija parametrizable

Irene Villar, Alberto Sanchez, Angel de Castro, Fernando López-Colino y Javier Garrido

***Resumen*—La verificación de reguladores digitales para convertidores conmutados no es trivial debido a la unión de un elemento analógico y otro digital. Aunque hay soluciones que permiten esta verificación, no hay una metodología estándar y, en muchos casos, la verificación es un proceso lento debido a las herramientas que se usan. Una alternativa es utilizar un sistema HIL (*Hardware-in-the-loop*) en el que se emula en hardware un modelo digital de la planta, generando simulaciones significativamente más rápidas. El objetivo de este artículo es implementar un modelo de un convertidor *full-bridge* utilizando aritmética en coma fija parametrizable, para integrarlo como sistema HIL. La implementación en coma fija, en vez de coma flotante, permite velocidades de emulación mayores, mientras que el área ocupada se mantiene por debajo del 2% en una FPGA (*Field-Programmable Gate Array*) Xilinx Zynq. El artículo muestra el diseño del modelo HIL, así como las pruebas que se realizan al modelo en lazo abierto, y también integrado en lazo cerrado y actuando como un inversor. En los resultados se muestra la precisión y los tiempos de simulación/emulación del modelo, así como los recursos utilizados en la FPGA.**

I. Introducción

El control digital para convertidores conmutados ha sufrido un gran crecimiento ya que ofrece beneficios significativos sobre el control analógico. Sin embargo, la verificación de un control digital junto a una planta analógica no es una tarea trivial. La realización de pruebas en el hardware real puede provocar accidentes debido a la gran energía que es capaz de manejar un convertidor conmutado. Por tanto, para el desarrollo de un control digital que regula una planta analógica, son necesarias simulaciones previas para comprobar su correcto funcionamiento. Sin embargo, no hay una metodología estándar para realizar estas pruebas, sino que en la literatura se encuentran diferentes alternativas.

A la hora de diseñar el regulador digital, se pueden utilizar herramientas como Matlab para comprobar la estabilidad, la respuesta, etc., pero cuando se implementa el regulador en hardware o software, pueden cometerse errores de implementación, difíciles de detectar.

Una opción para la depuración sería utilizar un simulador mixto digital-analógico, para poder simular planta y control simultáneamente [1, 2]. Sin embargo, el principal problema de estos simuladores es que la velocidad de simulación de estas herramientas es muy lenta. Por otro lado, los simuladores mixtos no son tan accesibles en términos de número de simuladores disponibles y precio respecto a otro tipo de simuladores. Además, estos simuladores no podrán probar la configuración final del sistema con los dispositivos finales, sus alimentaciones, interconexiones entre la unidad de control (microprocesador, DSP, etc) y ADCs (Analog-to-Digital Controller), etc. Otra solución propuesta en la literatura es utilizar dos simuladores, uno para el control y otro para la planta, y desarrollar una interfaz personalizada de comunicación entre simuladores [3], aunque actualmente ya existen simuladores comerciales con capacidad de interconexión.

Otra alternativa propuesta es diseñar un modelo digital de la planta en un lenguaje HDL (*Hardware Description Language*), lo que permite simulaciones más rápidas [4, 5, 6]. Al realizar este diseño se puede simplificar ligeramente el modelo, ya que la finalidad de esta etapa de verificación no es tanto medir la estabilidad del sistema y su respuesta, sino comprobar que la implementación final del control es correcta. En el caso de un regulador digital escrito en HDL, se puede implementar el modelo y el regulador en una FPGA (*Field-Programmable Gate Array*), creando un sistema de emulación HIL (*Hardware-in-the-loop*). Incluso si el control digital no está diseñado en HDL, sino en un microprocesador o, se puede emular el modelo de la planta en una FPGA y extraer los datos como señales analógicas utilizando DACs (Digital-to-Analog Converter), e interconectar las entradas y salidas del modelo HIL al dispositivo regulador. De esa forma, se puede probar la configuración final del sistema.

En la literatura se encuentran modelos HDL para convertidores conmutados en coma flotante y en coma fija. Los modelos en coma flotante de 32 bits son fáciles de implementar [9], pero no tienen suficiente resolución si el regulador tiene una frecuencia de conmutación alta (centenas de kHz) [10]. Se pueden usar más bits para aumentar la resolución, pero el área ocupada aumenta significativamente y la frecuencia de trabajo del modelo se ve reducida. Otra opción es diseñar un modelo en coma fija, asignando el número de bits necesarios para cada variable del modelo, y optimizando área y frecuencia de trabajo, pero aumentando el esfuerzo de diseño. Además, otro problema de los modelos en coma fija presentados en la literatura es que están definidos para unos rangos de valores fijos, es decir, sólo funcionan correctamente hasta unos niveles máximos de tensión, corriente, frecuencia de conmutación, etc. Esta restricción a la hora de diseñar, que en coma flotante no existe, puede paliarse si se realiza un modelo en coma fija parametrizable, donde el número de bits dedicados a la parte

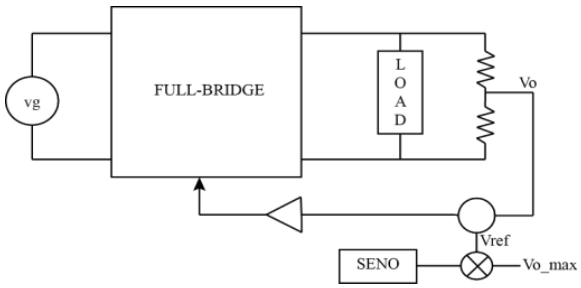


Figura 1. *Full-bridge* con control.

entera y parte fraccionaria sea variable.

Este artículo presenta cómo realizar un modelo en VHDL en coma fija parametrizable de un convertidor *full-bridge*, para su emulación HIL junto a un regulador.

La sección II detalla cómo crear un modelo del convertidor. La sección III explica los detalles de la implementación realizada. La sección IV muestra los resultados obtenidos y, por último, la sección V muestra las conclusiones.

II. MODELADO DEL CONVERTIDOR

Este artículo presenta la implementación de un modelo digital y parametrizable en coma fija de un convertidor *full-bridge*. Aunque se presente esta topología como ejemplo, el desarrollo de un modelo para otra topología usaría la misma metodología. Como ejemplo de aplicación, se va a probar el modelo junto a un control sencillo actuando como inversor (Figura 1). El sistema completo consta del modelo del convertidor *full-bridge*, un módulo seno para generar la referencia sinusoidal de la tensión de salida, y un regulador para el control de dicha tensión de salida.

A. Posibilidades de implementación

Si se opta por modelar la planta utilizando un lenguaje HDL, se pueden destacar cuatro posibilidades:

1. Modelo en coma flotante no sintetizable (utilizando señales de tipo *real* en el caso de VHDL): Es un modelo fácil de implementar ya que las señales ajustan automáticamente su exponente consiguiendo así la mejor resolución posible. Está soportado por la mayoría de los simuladores. Sin embargo, no puede ser sintetizado, por lo que no se puede incluir como modelo HIL. El tipo de datos *real* utiliza el estándar IEEE 754 de doble precisión (53 bits de mantisa), por lo que la resolución obtenida permite sobradamente el modelado de una planta incluso con frecuencias de conmutación elevadas.
2. Modelo en coma flotante sintetizable: Existen bibliotecas como *float* en VHDL que implementan este estándar. El estándar IEEE 754 propone dos formatos, precisión simple y doble precisión, donde la última exige una cantidad de recursos no asumible en muchos casos. En la literatura se encuentran modelos de convertidores usando coma flotante sintetizable de precisión simple [9], pero esta aritmética puede no tener suficiente resolución si la frecuencia de conmutación es alta y por tanto no siempre es viable en el modelado de convertidores conmutados [10].

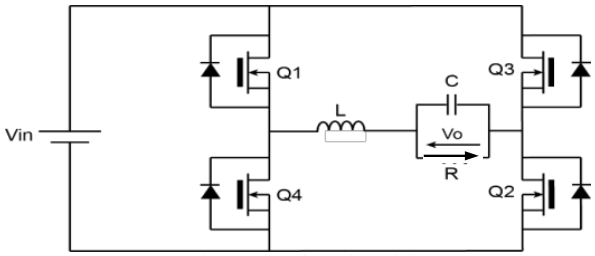


Figura 2. Topología de un *full-bridge*.

Aunque se puede aumentar el número de bits de la mantisa, en general la coma flotante requiere un uso enorme de recursos (10 veces mayor), mientras que la frecuencia es mucho menor (10 veces menor) [10].

1. Modelo en coma fija con librería *sfixed*: La coma fija permite obtener frecuencias de síntesis más altas utilizando menos recursos. La librería *sfixed* de VHDL facilita esta tarea interpretando los resultados en coma fija con sus respectivos tamaños y facilitando las operaciones aritméticas. Este modelado es sintetizable, pero debe ser diseñado teniendo en cuenta los rangos de valores que tendrá cada señal, quedando fijado, por tanto, el número de bits de parte entera y de parte fraccionaria y, consecuentemente, la máxima resolución. Por tanto, cambios en los rangos de valores reales durante la ejecución del modelo pueden hacer que la resolución sea insuficiente o que los valores desborden la capacidad total de una señal. Por ello, la complejidad de este modelo está en la elección de los tamaños de las señales.
2. Modelo en coma fija parametrizable: Se utilizan los fundamentos de la coma fija, pero sin utilizar ninguna biblioteca ya que a priori no se sabe el número de bits dedicados a las partes entera y fraccionaria. En el caso de VHDL se utiliza señales de tipo *std_logic_vector*, que no interpretan dónde está la coma, ni facilitan las operaciones aritméticas. Por tanto, las entradas y las salidas del modelo deben ser interpretadas externamente. Además, el modelo se realiza parametrizable de forma que, en vez de haber un número estático de decimales en cada señal, la posición de la coma puede cambiar de forma dinámica según la configuración de la simulación, y sin necesidad de resintetizar. Es decir, se consigue un modelo intermedio entre la coma flotante y la coma fija. Así, se trata de un modelo sintetizable, se pueden elegir los bits necesarios para cada señal de forma individual y la resolución siempre es óptima para todo rango de valores de la simulación. Esta configuración se puede hacer automáticamente ayudándose de una hoja de cálculo o un programa, donde únicamente habría que elegir las condiciones de la simulación (tensiones, corrientes, etc.) esperadas y calculándose, por tanto, el número de bits necesarios para cada parte.

El objetivo en este artículo es la implementación de un modelo parametrizable de un convertidor *full-bridge*, esto es, hacer un modelo genérico que funcione para cualquier valor que puedan tener las señales de entrada.

B. Modelado de la planta

La figura 2 muestra la configuración básica de un *full-bridge*. Está compuesto por cuatro interruptores con un diodo en antiparalelo en cada interruptor, dos elementos pasivos que son una bobina y un condensador y una carga que en este caso se ha considerado resistiva, aunque otro tipo de carga se puede modelar de forma similar.

Se trata de un circuito que puede actuar como inversor en función de la secuencia de apertura y cierre de los interruptores. El convertidor debe calcular en cada instante sus variables de estado, es decir, las señales que no pueden cambiar de valor de forma arbitraria en tiempo nulo. En el caso de un *full*-bridge,
las variables de estado son la tensión de salida (v_{out}) y la corriente que atraviesa la bobina (i_L).

La tensión de la bobina viene definida por (1)

$$v_L = L \cdot \frac{di}{dt} \tag{1}$$

Transformando (1) en una ecuación en diferencias en el instante *k,* la corriente de la bobina viene definida por (2)

$$i_L(k) = i_L(k-1) + \frac{\Delta t}{L} \cdot v_L \tag{2}$$

donde Δt es el paso de integración para calcular las variables de estado. De la misma manera, la corriente que atraviesa el condensador se va a definir como (3)

$$i_C = C \cdot \frac{dv}{dt} \tag{3}$$

Esta ecuación también se puede transformar en una ecuación en diferencias en el instante *k* (4)

$$v_{out}(k) = v_{out}(k-1) + \frac{\Delta t}{C} \cdot i_C \tag{4}$$

Se van a distinguir diferentes casos en función del estado de cada uno de los interruptores. Cuando los interruptores Q1 y Q2 están cerrados y Q3 y Q4 abiertos (5) se denomina rama 1, en la que la tensión de la bobina será la diferencia entre la tensión de entrada y la de salida $v_L = v_g - v_{out}$. En el caso contrario (6), Q3 y Q4 cerrados y Q1 y Q2 abiertos, siendo esta la rama 2, la tensión de la bobina es igual pero la tensión de entrada es negativa $-v_g - v_{out}$. Para el cálculo de la tensión de salida no afecta la tensión de entrada, por lo que, independientemente del estado de los interruptores, dependerá únicamente de la corriente que atraviesa el condensador $i_C = i_L - i_R$.

$$\begin{aligned} i_L(k) &= i_L(k-1) + \frac{\Delta t}{L} \cdot \left(v_g - v_{out}\right) \\ v_{out}(k) &= v_{out}(k-1) + \frac{\Delta t}{C} \cdot (i_L - i_R) \end{aligned} \tag{5}$$

$$\begin{aligned} i_L(k) &= i_L(k-1) + \frac{\Delta t}{L} \cdot \left(-v_g - v_{out}\right) \\ v_{out}(k) &= v_{out}(k-1) + \frac{\Delta t}{C} \cdot (i_L - i_R) \end{aligned} \tag{6}$$

Ambas ramas no pueden estar activas simultáneamente ya que se produciría un cortocircuito. Para evitar que se produzcan cortocircuitos debido a los retrasos de conmutación, es habitual añadir en el control un tiempo de transición entre la activación de una rama a otra, llamado *deadtime*, en el que ambas ramas se desactivan. Cuando todos los interruptores están abiertos, la corriente que debería circular por los interruptores atraviesa los diodos en antiparalelo. En este caso, cuando $i_L > 0$ sus ecuaciones son las correspondientes a la rama 2 (6), y cuando $i_L < 0$ sus ecuaciones son las correspondientes a la rama 1 (5).

Una vez definido el modelo matemático, se debe implementar el modelo en VHDL, siguiendo algunas de las aritméticas anteriormente citadas. Al modelo se le pueden añadir pérdidas eléctricas para aumentar la similitud con la realidad agregando estos términos en las ecuaciones de las ramas (5) y (6), pero se muestra el modelo básico para facilitar la comprensión del artículo.

III. Implementación

En esta sección se describe el proceso de implementación de dos de los modelos descritos en la sección II.A: modelo en *real* y modelo en coma fija parametrizable.

Implementar el modelo *real* consiste en transcribir directamente las ecuaciones descritas en (5) y (6) en código VHDL. Para su implementación no es necesario un esfuerzo extra por parte del diseñador para conseguir una resolución óptima ya que estas señales implementan coma flotante IEEE 754 de doble precisión (64 bits) y el ajuste de la coma se hace automáticamente. Pero como se ha visto anteriormente, aunque esté soportado por la mayoría de los simuladores, no es sintetizable. Debido a su gran resolución y, por tanto, la exactitud de sus resultados, se considerará como referencia para compararlo con el modelo en coma fija parametrizable.

El modelo en coma fija parametrizable utiliza la notación QX.Y. Este formato contiene 1 bit que permite signo, X bits de parte entera, Y bits de parte fraccionaria, considerándose escrito en complemento a 2, por lo que una señal Q2.3 tendrá 1+2+3 bits. Para obtener el valor decimal de un número en binario hay que tener en cuenta la *escala de la señal*, siendo esta la cantidad de bits que tiene la parte fraccionaria. Así, si se traduce una señal a decimal habría que multiplicar la señal por 2^{-Y}. Por ejemplo, si se tiene una señal que representa una tensión de salida y su valor binario es 1010010110100000_2 (21200 en decimal) con escala igual a 11, la tensión tendrá un valor de $21200 \cdot 2^{-11} = 10{,}3515625$ V.

Por otra parte, se trata de un modelo parametrizable, es decir, que se adaptará para cualquier dato de entrada. Para que esto sea posible, la escala de cada una de las entradas debe adaptarse en función del tamaño y precisión necesarios. La escala, o cantidad de bits de la parte fraccionaria, se obtiene de manera que se resta al número total de bits, el menor entero mayor del logaritmo en base 2 del valor que se quiere representar. Por ejemplo, si la simulación debe soportar tensiones de entrada de hasta 20 V y estando dicha señal definida con 16 bits, la escala será $16 - \lceil \log_2 20 \rceil = 11$. Sin embargo, si debe soportar 1000 V, la escala será $16 - \lceil \log_2 1000 \rceil = 6$. Por tanto, las resoluciones serán 2^{-11} y 2^{-6} V respectivamente, manteniendo una resolución aproximadamente proporcional al valor máximo soportado. Antes de realizar una simulación, pero sin necesidad de resintetizar el código, es necesario calcular las escalas de cada señal.

La figura 3 muestra el esquemático del *full-bridge* para el caso del modelado en formato coma fija parametrizable. Se pueden diferenciar dos partes: cálculo de la corriente de la bobina y cálculo de la tensión de salida. La primera parte consta de un

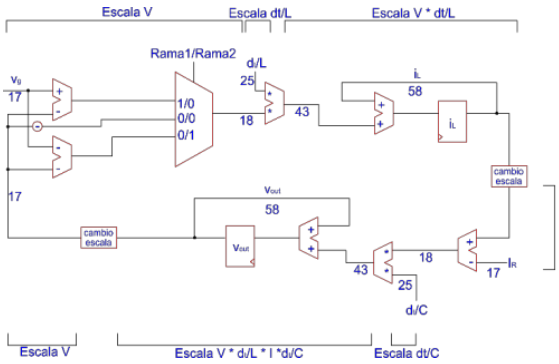


Figura 3. Implementación del modelo en coma fija parametrizable

multiplexor que, en función del estado de los transistores, elige cuál es la tensión aplicada en la bobina. Posteriormente se multiplica por $\frac{\Delta t}{L}$ y, finalmente, se acumula este incremento con la corriente del ciclo anterior. Este proceso se realiza de forma análoga para el cálculo de la tensión de salida.

En la figura 3 se pueden observar los tamaños de cada señal del modelo, pero su interpretación varía dependiendo de la escala de cada señal, también mostrada en la figura en forma de variables (V, dt/L, I, dt/C). Como se ha comentado, estas escalas dependen de los valores máximos esperados en las tensiones, tiempo de integración requerido (que depende de la frecuencia de conmutación), valor de la bobina, valor máximo esperado de la corriente y valor del condensador. El modelo se realimenta, así que no se pueden acumular las escalas, y es necesario realizar cambios de escala para poder operar correctamente con los datos de entrada. Para este fin, la figura 3 muestra dos bloques de cambio de escala. Las entradas numéricas del modelo (Vg e Ir) deben estar representadas en las escalas requeridas, y las salidas deben interpretarse dependiendo de las correspondientes escalas.

Las señales de salida del modelo se extraen mediante DACs para que un sistema de control externo las pueda leer. Estas salidas pueden adaptarse dinámicamente a los requerimientos del control (ganancias, offsets, etc).

IV. Resultados

Para comprobar la viabilidad del modelo parametrizable propuesto se deben realizar pruebas de precisión y de velocidad de simulación. Para dichos resultados, se compararán las simulaciones del modelo en coma fija parametrizable con las del modelo en coma flotante *real*. Este último modelo se puede considerar como referencia ya que usa el estándar IEEE-754 de doble precisión, con 53 bits de mantisa, y su resolución es muy alta incluso para modelos con frecuencias de conmutación altas y, por tanto, tiempo de integración muy bajo. El modelo realizado del *full-bridge* considera los parámetros: L= 900 µH, C= 100 µF y R= 12 Ω.

Para la obtención de estos resultados se han comparado los dos modelos implementados en diferentes situaciones: en lazo abierto en régimen permanente y en lazo cerrado actuando como inversor. Los códigos se han implementado en VHDL para poder ser implementados en una FPGA Xilinx xc7z020-1, mientras que los resultados de simulación se han obtenido con la herramienta ModelSim 10.0b. Aunque se van a mostrar datos extraídos de simulaciones VHDL, los resultados reales son extraídos, como se ha comentado mediante DACs de alto ancho de banda y 14 bits de resolución, siendo iguales salvo por los pequeños retrasos del citado DAC y su bajo error de resolución.

En lazo abierto se ha definido una tensión de entrada Vin = 20 V, y un ciclo PWM invariable, por lo que se puede comprobar si realmente la planta en coma fija parametrizable se comporta de forma similar al modelo *real*. Como no se está probando un control en lazo cerrado, si la planta en coma fija sufre un error, no habrá regulador que lo corrija, pudiéndose analizar el error con facilidad comprobando los valores de las variables de estado. En la figura 4 se muestra la superposición de los resultados de la tensión de salida del modelo real y en coma fija con un ciclo de trabajo de 0,1. Como se puede ver, la respuesta es similar. Debido a la exactitud de los resultados con los de coma flotante, la figura 5 muestra una ampliación de la gráfica anterior donde se aprecian las diferencias. En la figura 5 se puede observar un comportamiento discreto en los valores de coma fija. Esto es porque en esa ampliación, la tensión de salida está cambiando con incrementos tan pequeños similares a la resolución máxima que se obtiene con 58 bits y, por tanto, los

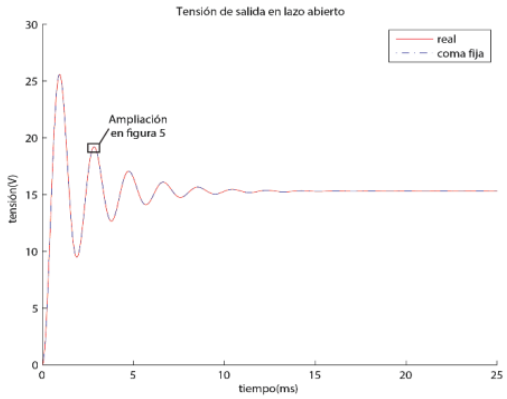


Figura 4. Comparación de tensiones de salida en lazo abierto con ciclo de trabajo de 0,1.

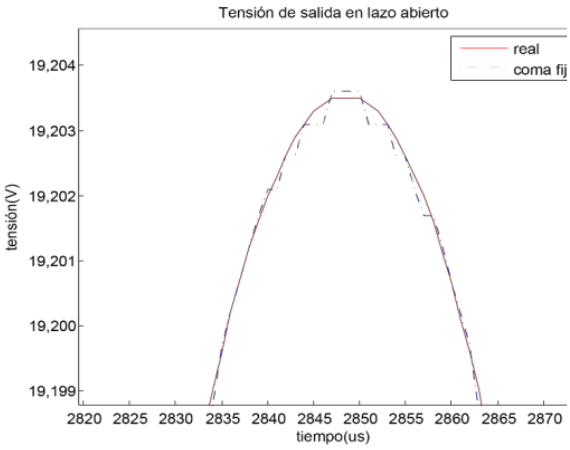


Figura 5. Aplicación de la comparación de tensiones de salida en lazo abierto.

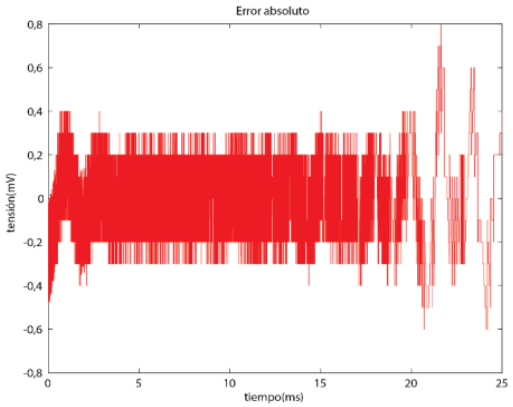


Figura 6. Error absoluto de las tensiones de salida en lazo abierto.

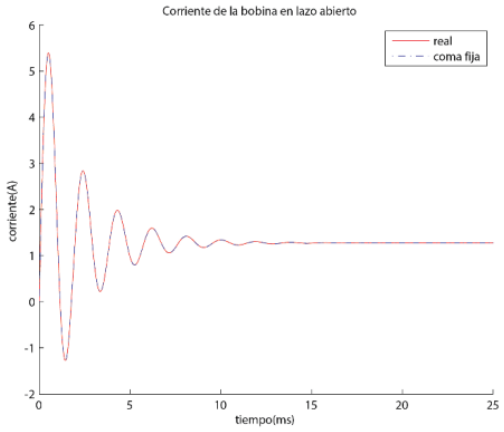


Figura 7. Comparación de corrientes de la bobina en lazo abierto con ciclo de trabajo de 0,1.

escalones que se pueden observar es el salto de un valor binario a justo el siguiente. Es decir, para las escalas elegidas para este ejemplo, la resolución está en torno a 0,6 mV.

La figura 6 muestra la gráfica del error absoluto de la figura anterior (figura 5). El error aumenta cuando la figura 4 está en régimen permanente, a los 20 ms aproximadamente, llegando incluso a duplicar su error. Esto es debido a la resolución del modelo en coma fija. Cuando el modelo está en régimen permanente, el rizado de la tensión de salida está siendo tan pequeño que, nuevamente, se acerca a la resolución máxima admitida para el modelo con las escalas elegidas. Se puede observar que la tensión de salida tiene un error absoluto medio de $1{,}54 \cdot 10^{-4}$ V con una desviación típica de $1{,}19 \cdot 10^{-4}$ V.

Al igual que se han comparado las tensiones de salidas, la figura 7 muestra la comparación de las corrientes de la bobina. Como se puede observar, los resultados son similares para la corriente. Por tanto, el valor que se ha elegido para el resto de las pruebas ha sido la tensión de salida.

La prueba anterior demuestra que el modelo en coma fija, a pesar de utilizar un número mucho menor de bits para las variables de estado, obtiene errores de resolución de muy bajo

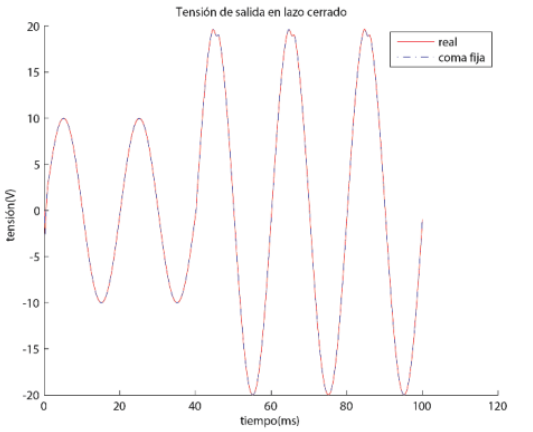


Figura 8. Comparación de tensiones de salida en lazo cerrado con distintos valores de tensión de referencia, 10 V y 20 V.

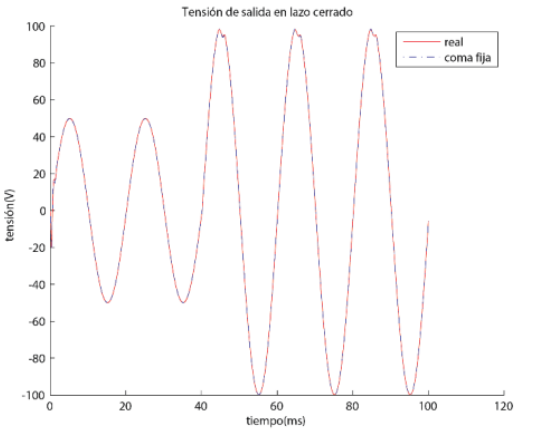


Figura 9. Comparación de tensiones de salida en lazo cerrado con distintos valores de tensión de referencia, 50 V y 100 V.

valor, por lo que dicho modelo se comporta de forma similar al modelo de referencia *real*.

A continuación se muestran las pruebas en lazo cerrado actuando como inversor, es decir, para el sistema completo incluyendo un regulador y una referencia de tensión sinusoidal.

Se ha realizado una simulación con un transitorio con distintos valores de referencia siendo estos 10 V y 20 V con tensión de entrada igual a 20 V (figura 8), para un primer caso, y 50 V y 100 V, con tensión de entrada igual a 100 V (figura 9) para un segundo caso.

Al igual que en lazo abierto, la exactitud de los resultados impide diferenciar a simple vista un modelo de otro, por lo que los errores debido a la pérdida de resolución son muy pequeños. El error absoluto medio en el primer caso (20 V de tensión de entrada) es de 0,0047 V y en el segundo caso (100 V de tensión de entrada) es 0,072 V, con una desviación típica de 0,0035 V y 0,1824 V respectivamente. Una vez se ha visto la precisión del modelo parametrizable en distintos casos, se procede a comprobar los tiempos de simulación y emulación, ya que el principal objetivo de un sistema HIL es aumentar la velocidad

TABLA I
RESULTADOS DE TIEMPO DE SIMULACIÓN DE 100 MS

Sistema	Simulación/Emulación	Tiempo
Tipo real	Simulación	54" 120 ms
Coma fija parametrizable	Simulación	1' 27" 70 ms
Coma fija parametrizable	Emulación	255,6 ms

TABLA II
RECURSOS USADOS EN EL DISEÑO EN UNA FPGA XILINX XC7Z020-1

Sistema	Máx. Frec.	LUTs		FFs		Mult	
		Usado	%	Usado	%	Usado	%
Modelo Coma flotante Float32	12,20 MHz	8987	16	69	1	4	1
Modelo Coma fija *sfixed*	46.84 MHz	415	1	109	0,1	3	1
Modelo Coma fija parametrizable	45,22 MHz	509	1,5	106	0,1	3	1

de verificación de un control digital. Para calcular estos tiempos se ha probado a realizar una simulación de 100 ms del sistema. La simulación se realizó en un ordenador con procesador Intel Core i7-4770 a 3,4 GHz y 8GB de RAM. La tabla I recoge estos resultados. En el caso del sistema emulado (ejecutado en una FPGA), para que la emulación se ejecute a tiempo real, el paso de integración tendría que ser igual al periodo de reloj de la FPGA. Como dicho valor, Δt, también es parametrizable, ya que puede variar dependiendo de la frecuencia de conmutación, el usuario podrá cambiarlo llegando a un compromiso entre velocidad de simulación y precisión en el modelo.

Por otro lado, la tabla II muestra los recursos utilizados por la FPGA Xilinx xc7z020-1 para implementar el sistema HIL utilizando diferentes aritméticas. Se puede observar la gran cantidad de recursos que utiliza la aritmética float32 y su baja frecuencia de funcionamiento. Esta frecuencia hace que el tiempo de integración del modelo sea mayor y, por tanto, su precisión disminuye. Los modelos en coma fija funcionan a frecuencias mucho más altas utilizando un número menor de recursos, por lo que varios modelos podrían implementarse en un mismo dispositivo HIL. La ventaja de coma fija parametrizable frente a *sfixed* es que la primera puede adaptarse frente a diferentes condiciones de simulación, mientras que un modelo *sfixed* puede desbordar sus señales o, en cambio, puede tener insuficiente resolución si dichas condiciones cambian.

V. CONCLUSIONES

En este artículo se han planteado distintos métodos de implementación para el modelado HIL de un convertidor *full-bridge* en HDL. El modelo en coma flotante no sintetizable (con señales tipo *real*) es muy sencillo de implementar y está soportado por la mayoría de los simuladores, sin embargo, no es sintetizable. El modelo en coma flotante sintetizable no es viable para convertidores conmutados debido a su falta de resolución cuando la frecuencia de conmutación es muy alta. La alternativa en coma fija con librería *sfixed*, aunque permite obtener frecuencias de síntesis más altas utilizando menos recursos, su resolución puede ser insuficiente si se producen cambios en los rangos de valores durante la ejecución del modelo. Por último, en el artículo se ha presentado un modelo en coma fija parametrizable, que posee las propiedades de la coma fija pero que, a la vez, se puede adaptar a cualquier rango de valores, ya que la coma es dinámica sin necesidad de resintetizar el modelo.

Para este modelo se ha comprobado tanto su precisión como sus tiempos de simulación respecto al modelo con señales de tipo *real*. Con los resultados se puede concluir que el modelo en coma fija parametrizable es un modelo de gran precisión, ya que sus errores absolutos con el modelo real son de centenas de microvoltios cuando la tensión de salida está en el orden de las decenas de voltios. Además, se ha comprobado que la emulación en FPGA de este modelo permite verificaciones rápidas, llegando a tiempo real si el periodo de integración está en el orden de las decenas de nanosegundos.

Referencias

[1] A. Prodic and D. Maksimovic, "Mixed-signal simulation of digitally controlled switching converters," in Proc. IEEE Workshop Comput. Power Electron., Mayaguez, PR, USA, Jun. 2002, pp. 100–105.

[2] L. Barragan, I. Urriza, D. Navarro, J. Artigas, J. Acero, and J. Burdio, "Comparing simulation alternatives of FPGA-based controllers for switching converters," in Proc. IEEE Int. Symp. Ind. Electron. (ISIE), Boulder, CO, USA, Jun. 2007, pp. 419–424.

[3] P. Zumel, M. García-Valderas, A. Lázaro, C. López-Ongil, and A. Barrado, "Co-simulation PSIM-ModelSim oriented to digitally controlled switching power converters," in Proc. IEEE 12th Workshop Control Model. Power Electron. (COMPEL), Jun. 2010, pp. 1–7.

[4] S. Karimi, P. Poure, and S. Saadate, "An HIL-based reconfigurable platform for design, implementation, and verification of electrical system digital controllers," IEEE Trans. Ind. Electron., vol. 57, no. 4, pp. 1226–1236, Apr. 2010.

[5] M. Matar and R. Iravani, "FPGA implementation of the power electronic converter model for real-time simulation of electromagnetic transients," IEEE Trans. Power Del., vol. 25, no. 2, pp. 852–860, Apr. 2010.

[6] Y. Chen and V. Dinavahi, "Digital hardware emulation of universal machine and universal line models for real-time electromagnetic transient simulation," IEEE Trans. Ind. Electron., vol. 59, no. 2, pp. 1300–1309, Feb. 2012

[7] E. Adžic, M. Adžic, V. Katic, D. Marčetic, and N. Ćelanovič, "Development of high-reliability EV and HEV IM propulsion drive with ultra-low latency HIL environment," IEEE Trans. Ind. Informat., vol. 9, no. 2, pp. 630–639, May 2013.

[8] A. Hasanzadeh, C. Edrington, N. Stroupe, and T. Bevis, "Real-time emulation of a high speed micro-turbine permanent magnet synchronous generator using multi-platform hardware-in-the-loop realization," IEEE Trans. Ind. Electron., vol. 61, no. 6, pp. 3109–3118, Jun. 2014.

[9] O. Lucia, I. Urriza, L. Barragan, D. Navarro, O. Jimenez, and J. Burdio, "Real-time FPGA-based hardware-in-the-loop simulation test bench applied to multiple-output power converters," IEEE Trans. Ind. Appl., vol. 47, no. 2, pp. 853–860, Mar.–Apr. 2011.

[10] A. Sanchez, A. de Castro & J. Garrido, "A comparison of simulation and hardware-in-the-loop alternatives for digital control of power converters", in IEEE Transactions on Industrial Informatics, vol. 8, no. 3, pp. 491-500, ago 2012.

Hardware-in-the-loop using parametrizable fixed point notation

Area of contribution: Modeling and simulation

Alberto Sanchez1, Irene Villar, Angel de Castro, Fernando López-Colino, Javier Garrido

Hardware and Control Technology Laboratory, Dpto. TEC, Escuela Politécnica Superior, Universidad Autónoma de Madrid

1Francisco Tomás y Valiente 11, C-231.
28049 Madrid (Spain).
Phone number: +34- 91 497 3614
Email: alberto.sanchezgonzalez@uam.es

Abstract

Keywords

Hardware-in-the-loop, Real time simulation, FPGA

Hardware-in-the-loop using parametrizable fixed point notation

I. Introduction

In recent years, the simulation of power converters using HIL (Hardware-in-the-Loop) systems is increasing, because of its speed-up compared to traditional simulation techniques. The idea is to implement the model of the plant in digital hardware, which can be a computer, a microprocessor, an FPGA, etc. The first proposals in the literature used computers, but their simulation time step was high and they were only useful for low frequency applications [1].

More recently, FPGAs have been introduced in HIL systems for acceleration. In [2-3], several HIL systems based on FPGAs model low switching frequency converters. In these proposals, fixed point numerical notation were used. Fixed point representation achieves optimal results in area and speed, but the design effort is increased compared to using floating point representation. In [4], float_pkg, which is a library inside VHDL2008, is proposed for HIL. This library helps to implement synthesizable floating points operations. Floating point helps the designer to model the plant, but its running frequency is around ten times slower, and the used area is more than ten times greater [5].

Another remarkable difference between fixed and floating point notations is the representation range of a number. In fixed point, the range is constrained at design time, while in floating point the location of the decimal point can be shifted when necessary, although the resolution of the mantissa is fixed (23+1 bits in IEEE-754 single precision). Therefore, the main drawback of fixed point is that the simulation values cannot exceed the range imposed in the design stage of the model.

This paper proposes the implementation of a HIL model based on parametrizable fixed point representation. This approach allows to configure the location of the decimal point at simulation time instead of design time, so it is not necessary to reimplement the model if the simulation conditions change. In this way, he advantages of both fixed point and floating point are obtained. The rest of the paper is organized as follows. Section II explains how to model a power converter plant. Section III describes its implementation. Section IV presents the results and, finally, conclusions are given in Section V.

J. II. Model of the plant

This paper presents a HIL system based on parametrizable fixed point notation using an FPGA. The application example is a full-bridge converter (see Fig. 1). There are several possibilities to model the plant in an FPGA:

1. Floating point representation: It allows small design time because the designer can transcribe the mathematical equations almost directly to VHDL, without taking into account the representation ranges of a number, because the point is shifted automatically when necessary. In [4] it can be found an HIL model using this 32-bit floating point representation (single precision standard). However, resolution problems have arisen in models used with high frequency switching [5] (hundreds of kHz). Double precision can be used but the results of area and speed dramatically deteriorate.
2. Fixed point representation: It allows smaller simulation steps and uses less area than floating point, but it requires more design effort. This is because every signal should be implemented taking into account the value ranges inside the simulation. The point location is fixed, so the number of bits for the integer and fractional parts are constrained at design time. A conservative decision would be to allocate many bits to the integer part to avoid numerical saturation/overflow, but the resolution would decrease.
3. Parametrizable fixed point representation: It uses the basis of fixed point representation, taking advantage of the high speed and low area. However, the number of bits for the integer and fractional parts are not known a priori. This model uses integer signals and the model is not aware about the point location. Thus, all inputs and outputs of the model are externally interpreted. Depending on the interpretation, the number of integer and fractional bits are changed so, given the simulation limits, the system adjusts the point locations of the signals without resynthesizing.

The full-bridge converter can be modeled analyzing its state variables, which are the output voltage (v_{out}), and the inductor current (i_L). The converter can be modeled implemented using Eq. 3 when top-left and bottom-right switches are closed, and Eq. 4 when top-right and bottom-left switches are closed:

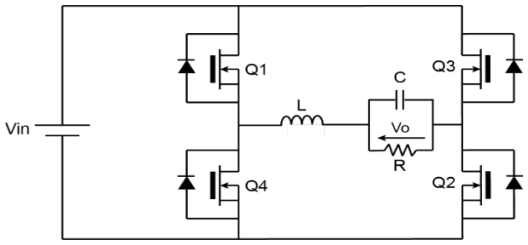


Figure 1. Schematic of a full-bridge converter.

$$i_L(k) = i_L(k-1) + \frac{\Delta t}{L} \cdot \left(v_g - v_{out}\right)$$
$$v_{out}(k) = v_{out}(k-1) + \frac{\Delta t}{C} \cdot (i_L - i_R) \quad (3)$$
$$i_L(k) = i_L(k-1) + \frac{\Delta t}{L} \cdot \left(-v_g - v_{out}\right)$$
$$v_{out}(k) = v_{out}(k-1) + \frac{\Delta t}{C} \cdot (i_L - i_R) \quad (4)$$

where k represents a simulation step, Δt is the simulation time step, L is the inductance, C the capacitance, v_L the voltage of the inductor and i_C the current through the capacitor. The extraction of the model equations will be explained with detail in the final manuscript.

III. Implementation

The proposed fixed point notation model uses QX.Y representation. This format contains 1 bit which allows sign, and X and Y bits for the integer and fractional parts respectively. To obtain the decimal value of a QX.Y number, it should be multiplied by 2-Y. However, the model is not aware of the value of X and Y, so it only operates the numbers as if they were integers.

The proposed system is parametrizable, so the scale (value of Y) of every signal should be adapted to the needed value range. The scale of a signal can be obtained with Eq. 5:

$$Y = Total\_number\_of\_bits - \lceil \log_2 Max\_value \rceil \quad (5)$$

It can be seen in Eq. 5 that the resolution is adjusted depending on the value range, improving the versatility of the classic fixed point notation. The scale of the signals should be calculated before simulating in order to send to the model the input values and interpretate its output values. However, this can be done without resynthesizing.

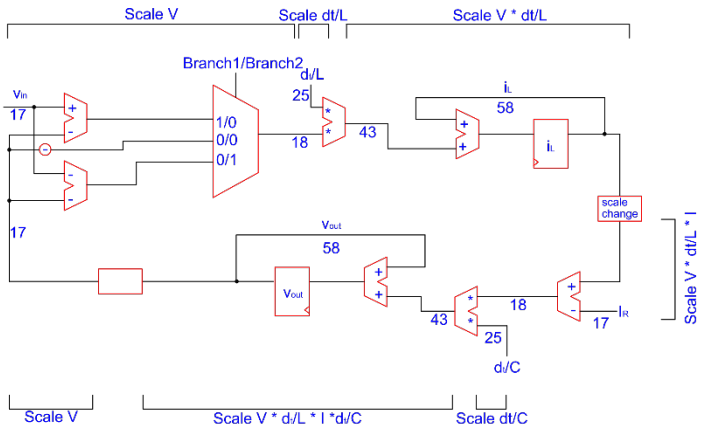


Figure 2. Implementation of the parametrizable fixed point model.

Fig. 2 shows the schematic of the model of the full-bridge converter using this approach. Two parts can be seen: the calculus of the inductor current and the output voltage. The total number of bits of every signal and their scales are shown in the figure. As there is a feedback in the model (there is dependency between current and voltage), the internal variables (in their own scale) are shifted when passed to the other part of the model, so both operands in the adders/subtractors are in the same scale. More details will be shown in the final manuscript.

IV. Results

The parametrizable fixed point approach has been implemented using a FPGA Xilinx xc7z020-1. The proposal has been compared with a non-synthesizable 64-bit floating point model, which resolution is sufficiently high, so it can be considered as a reference system. The physical parameters of the model are Vin= 20 V, L= 900 µH, C= 100 µF, R= 12 Ω, and a simulation step of 25 ns.

The model has been simulated using a fixed PWM (duty cycle = 0.1), instead of closed loop, as the regulator would minimize the error of the converter model. Fig. 3a shows the results of the output voltage for the 64-bit floating point model (called real), and the parametrizable fixed point proposal. As the model output is similar in both cases, Fig. 3b shows a detail of the output voltage so both systems can be compared. It can be seen a discrete behavior in the fixed point model. This is because 0.6 mV is the resolution of the output voltage signal in this example, so there the minimum step has this magnitude. As it was explained before, the resolution is adapted depending on the maximum representable value. The average error of the fixed point model is $1.54 \cdot 10^{-4}$ V with a standard deviation of $1.19 \cdot 10^{-4}$ V. More results in closed loop with a regulator will be found in the final manuscript.

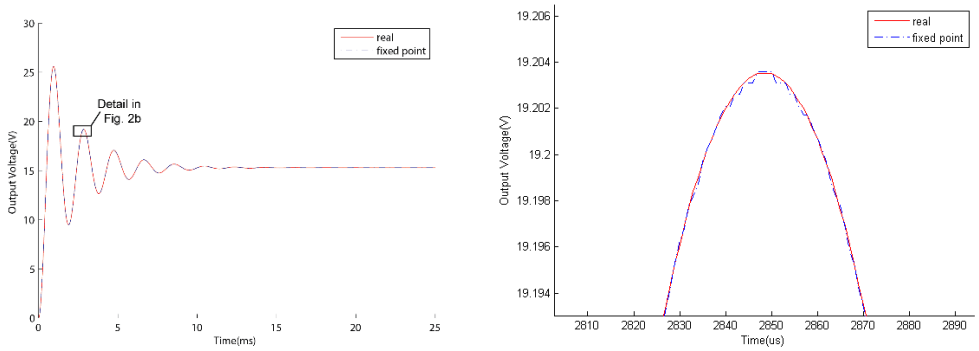


Figure 2. a) Output voltage results in open loop. b) Detail zoom in.

V. Conclusion

This paper has proposed a hardware-in-the-loop system to emulate a power converter using parametrizable fixed point notation. The proposals found in the literature use floating point and fixed point arithmetics. 32-bit floating point is versatile and allows rapid design, but the resolution may not be enough for high frequency switching converters. Using more bits, the resolution problem can be avoided, but the results in area and speed may not be affordable. Fixed point notation is more complex but every signal has the optimal number of bits. The main drawback of fixed point is that the simulation cannot exceed the design value limits. The proposed parametrizable fixed point notation achieves the versatility of floating point and the speed and area of traditional fixed point arithmetic. Results show that this proposal achieves almost the same numerical results, whilst the area is kept low and it achieves real-time emulation with a simulation step of 25 ns.

References

[1] B. Lu, X. Wu, H. Figueroa, and A. Monti, “A low-cost real-time hardware-in-the-loop testing approach of power electronics controls,” IEEE Trans. Ind. Electron., vol. 54, no. 2, pp. 919–931, Apr. 2007.

[2] G. Parma and V. Dinavahi, “Real-time digital hardware simulation of power electronics and drives,” IEEE Trans. Power Delivery, vol. 22, no. 2, pp. 1235–1246, Apr. 2007.

[3] Y. Chen and V. Dinavahi, “Digital hardware emulation of universal machine and universal line models for real-time electromagnetic transient simulation,” IEEE Trans. Ind. Electron., vol. 59, no. 2, pp. 1300–1309, Feb. 2012.

[4] O. Lucia, I. Urriza, L. Barragan, D. Navarro, O. Jimenez, and J. Burdio, “Real-time FPGA-based hardware-in-the-loop simulation test bench applied to multiple-output power converters,” IEEE Trans. Ind. Appl., vol. 47, no. 2, pp. 853–860, Mar.–Apr. 2011.

[5] Left blank for blind review.